ESD-TR-76-354



# MULTICS SECURITY INTEGRATION REQUIREMENTS
1 January 1976 - 31 December 1980

Honeywell Information Systems, Incorporated
Federal Systems Operations
McLean, VA 22101

March 1976



Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

## LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

## OTHER NOTICES

Do not return this copy. Retain or destroy.

This technical report has been reviewed and is approved for publication.

William R Price

WILLIAM R. PRICE, Capt, USAF
Techniques Engineering Division

Donald P. Eriksen

DONALD P. ERIKSEN
Techniques Engineering Division

FOR THE COMMANDER

Stanley P. Dereska

STANLEY P. DERESKA, Colonel, USAF
Deputy Director, Computer
Systems Engineering
Deputy for Command & Management Systems

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>ESD-TR-76-354 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>MULTICS SECURITY INTEGRATION REQUIREMENTS<br>1 January 1976 - 31 December 1980 | | 5. TYPE OF REPORT & PERIOD COVERED<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>N. Adleman, R. Ziller, J. Whitmore, et al | | 8. CONTRACT OR GRANT NUMBER(s)<br>F19628-74-C-0193 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Honeywell Information Systems, Inc.<br>Federal Systems Operations<br>McLean, VA 22101 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS<br>CDRL A006 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Deputy for Command and Management Systems<br>Electronic Systems Division, Hanscom AFB, MA 01731 | | 12. REPORT DATE<br>March 1976<br>13. NUMBER OF PAGES |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Security
Multics
Security Kernel
Certification
Secure Front-End Processor
Multilevel Security

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report presents the results of a study to identify a technical approach for the design, development and certification of a secure general purpose computer system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PROJECT GUARDIAN

# Multics Security Integration Requirements

1 January 1976 - 31 December 1980

Prepared for

Department of the Air Force

Electronic Systems Division

Hanscom Air Force Base

Bedford, Massachusetts 01731

Contract F19628-74-C-0193

Honeywell Information Systems, Inc.
Federal Systems Operations
McLean, Virginia

# Preface

This report is the result of a study to identify a technical approach for the design, development and certification of a prototype secure general-purpose computer system. The documented approach including the example schedules and resource estimates does not represent an approved Air Force program but merely describes one feasible structuring of the project with associated estimates of the magnitude of effort required.

Contents

# Introduction

## Introduction To The Integration Plan

The problem of security in computer systems has been under study for several years. The Air Force has sponsored studies and development projects aimed at improving understanding of security in computer systems, developing a sound theoretical basis for further work, and demonstrating accomplishments in the field. Many of these projects have been associated with the Multics system.

The overall goal of these efforts has been to develop a certifiably secure computer system for general use by the military to meet their operational requirements. This report presents a plan to take the Multics system from its present form to a prototype Multics system which can be used to demonstrate the feasibility of software certification. As part of this effort, a Secure Front-End Processor (SFEP) will be developed for eventual integration and certification. Another activity will be the identification of the technology and of a set of tools which will allow eventual certification of the Multics and SFEP software kernels.

### Background

The military faces an increasing need for operational computer systems capable of processing several levels of classified information at the same time. Present systems (except Multics) are unable to support secure multilevel processing due to fundamental weaknesses in their basic design since security was not a concern when they were developed. The weakness is that current hardware/software systems are unable to adequately protect the information that they process.

With the exception of the two-level Multics system developed for, and in use at, the Air Force Data Services Center, the military currently meets the need for processing several levels of information by one of two methods. Either all security levels are processed together at the level of the highest classification present, or each level is processed by itself. Both methods have been less than satisfactory. The problem with processing all levels together is that all users and all equipment, including terminals and communications facilities, must be cleared to the highest classification that the system can ever process. The problem with separate processing is that a separate computer system or a separate period of time is required for each level handled. Either method is costly and inefficient. Neither method allows simultaneous handling of information at several levels for users of several levels of clearance.

Multics is the most advanced general computing system as far as security is concerned. Security was one of the initial design goals of the Multics system designers and has been a major concern of the designers and developers throughout the history of the system. Even with this concern for security, the present Multics system cannot be certified secure. Multics, however, does present the best available base upon which to build a certifiably secure multilevel computer utility.

Secure communications has also presented operational problems to the military. A secure on-line system requires a secure communications network. While the techniques of securing communications lines and terminals have been well developed, a certifiably secure communications processor is still undeveloped. A secure multilevel system must have a compatible and secure front-end communications processor to be able to properly handle multiple levels of classified information.

Both economic and operational considerations make development of a certifiably secure multilevel system desirable. Recent advances in computer technology indicate that it should be possible to produce a system that can process an arbitrary mix of classified and unclassified information simultaneously on a single computer system. The system should serve both cleared and uncleared users and should rely on the computer system's internal hardware/software controls to enforce security and need-to-know requirements. Of primary importance is that the system be certifiably secure. That is, it must be possible to prove that the system is complete and without flaw in any of its security-related aspects.

The Air Force has been working on the problem of providing a certifiably secure multilevel system. In 1970, the Air Force Data Services Center (AFDSC) requested the Electronic Systems Division (ESD) to support development of an open multilevel system for the AFDSC Honeywell 635 systems. The resulting studies pointed out the severity of the problem and led to the formation of a computer security technology planning study panel. The panel's report (1) described the fundamental problems and delineated a program to develop the desired system. The panel recommended that the technical approach to the problem be "to start with a statement of an ideal system, a model, and to refine and move the statement through various levels of design into the mechanism that implements the model system".

The basic component of the ideal system was also identified by this panel. This component is known as the Reference Monitor, an abstract mechanism that controls access of subjects (active system elements) to objects (units of information) within the computer system and enforces the rules of the military security system on such access. Three requirements were recognized for a Reference Monitor:

1. Complete Mediation - the mechanism must mediate every access of a subject to an object.

2. Isolation - the mechanism and its data bases must be protected from unauthorized alteration.

3. Verifiability - the mechanism must be small, simple, and understandable so that it can be completely tested and verified (certified) to perform its functions correctly.

The mechanism that implements the Reference Monitor in a particular computer system has been termed the security kernel. Much subsequent work has been devoted to identifying the characteristics of a security kernel and to exploring the technology involved in producing a security kernel for some computer system.

ESD initiated development of formal mathematical models of the ideal Reference Monitor in 1972. This work (2,3) resulted in a model of a secure computer system as a finite-state mechanism that makes explicit transitions from one secure state to another. The rules of the model formally define the conditions under which a transition from state to state can occur. The rules have been proven to allow only transitions that preserve the security of information in the system. The model specifies requirements for the operation of a security kernel. These requirements were taken directly from the Defense Department regulations on handling sensitive information (DoD Directive 5200.1-R). With the availability of the model, the problem of validation is now reduced to providing complete assurance that a particular security kernel behaves exactly as the model requires.

Work on the technology of certification progressed in parallel with the work on the model. In 1973, W. Price (4) identified a methodology for verification of a kernel. More detailed developments of this validation methodology have been reported by MITRE (5,6). Another approach has been explored which may be more suitable to large software modules (7).

Other activities have been devoted to the problem of building a security kernel for a practical system (8,9). This work has demonstrated the soundness of the basic concepts and also pointed out some of the problems that lie in the way of realizing a security kernel on a large system. This work has been the basis for development of a secure communications processor, a detailed effort which is presented later in this report.

A major project in the development process is the development of a security kernel for a large resource sharing system. The system chosen for this effort is Multics. There are two reasons that this choice was made. First, the hardware base of the Multics system, the Honeywell 68/80 computer, has been identified

as best suited of all off-the-shelf large computer systems for the support of a security kernel (10). Second, the Multics system architecture was conceived and developed with security requirements specifically in mind.

One project, now completed, involved the design and production of a Multics system capable of supporting a two-level (Secret and Top Secret) environment for the Air Force Data Services Center (11,12). This system implements security controls based on the military access rules, but it does not completely handle the threat of a hostile penetration. From these efforts, additional insight was gained in the problems of designing and developing a security kernel for Multics.

## Summary of Integration Plan

Design of a security kernel for Multics was started as a joint effort among personnel from ESD, the MITRE Corporation, the Massachusetts Institute of Technology, and Honeywell Information Systems. This design effort has led to a more complete understanding of the general problem and has provided the foundation for the development plan which is presented in the following sections of this report. Work is progressing on formal specifications for the Multics security kernel (13) and on simplification and reorganization of the Multics operating system. Based on these efforts, this report describes a major project to refine the current Multics system and reimplement critical portions of the system to produce a certifiable kernel which will interface with a certifiable front-end communications processor with its own security kernel. The result will be a prototype Multics system which may meet the goal of Air Force certification.

The project can be described in terms of three coordinated development activities: development of hardware for a prototype Secure Front-End Processor (SFEP) and Interface Units; development of a security kernel for the Secure Front-End Processor; and development of a certifiable prototype secure Multics including the Secure Front-End Processor.

Programming methodologies and techniques will be selected to support the major Multics and SFEP development activities. Included are: techniques for formally specifying software; techniques for demonstrating the correspondence among hierarchically ordered levels of specification; programming languages (or subsets of programming languages) emphasizing the generation of certifiable code; support tools aiding the task of certifying both specifications and code; and techniques for performance measurements.

The Secure Front-End Processor is developed using a hardware architecture designed specifically to provide a basis for

certification according to the Air Force security model. The hardware provides segmented addressing and interfaces directly with the prototype Multics system. The software provides a kernel-based system architecture with a supervisor supporting communications subsystems for external I/O. The initial version of the SFEP software is integrated with the kernel-based Multics system to demonstrate functional capability. A second SFEP version is then coded in the selected system programming language, incorporates performance enhancements and is then integrated with Multics.

Development of the certifiable prototype Multics system entails the restructuring of the present Multics supervisor to rely on a formally specified security kernel leading to three demonstration systems. The preliminary Multics demonstration employs a formally specified kernel, coded in PL/I, interfacing with a DATANET 6600 front-end processor and establishes functional and performance measures of the design. The intermediate Multics demonstration incorporates the Secure Front-End Processor to demonstrate successful integration of the two hardware systems with their respective kernels and supervisors. The operational prototype Multics demonstration incorporates performance enhancements and contains kernels coded in the selected system programming language. This last system establishes the feasibility of certifying code correctness aided by the support tools developed as part of the project.

The kernels for the operational prototype Multics demonstration will be developed in a secure development facility. Upon successful conclusion and demonstration of the operational prototype secure Multics, the final product of this project is then available for installation for test and evaluation purposes at an operational government service site such as the Air Force Data Services Center (AFDSC). The actual installation and evaluation of the operational prototype secure Multics is beyond the scope of this five year plan. However, a conversion plan and approach to describe, achieve, and support that installation is provided in this Integration Plan.

This Integration Plan encompasses a five year effort during which major technical milestones will occur. The attached schedule of milestones shows the planned time frame for these major technical events which are described as follows:

1. SFEP Development Unit

   Delivery of SFEP system hardware to the hardware system test software development facility at Honeywell's Aerospace Division.

2. Preliminary Multics Demonstration

   Specification, design, implementation, and demonstration of the first kernel-based Multics system. This system is coded in PL/I and utilizes a DATANET-6600 communications processor.

3. SFEP Prototype System

   Specification, design, and assembly language implementation of a prototype SFEP software system.

4. TEMPEST Tested SFEP Hardware

   Availability of prototype SFEP hardware which conforms to TEMPEST requirements.

5. System Programming Language

   Availability of operational compilers which facilitate program correctness proofs to support the reimplementation of the Multics and SFEP kernels for the operational prototype Multics demonstration.

6. Intermediate Multics Demonstration

   Demonstration of an integrated, PL/I-coded, kernel-based Multics system with SFEP capabilities.

7. Tools for Correctness Proofs

   Selection of correctness proving tools to facilitate certification of the kernels in the operational Multics prototype.

8. Begin Correctness Proofs

   Begin feasibility demonstration for proof of correspondence between the specification and kernel software implementations.

9. Operational Prototype Multics Demonstration (Recoding and System Integration)

   Reimplementation of the security kernels in the selected system programming languages and implementation of performance enhancements.

10. Correspondence Technology and Specification Proofs

    Completion of the correspondence proofs between the model and the kernel specifications for the prototype using the available correspondence proof tools.

11. Establish Secure Development Site

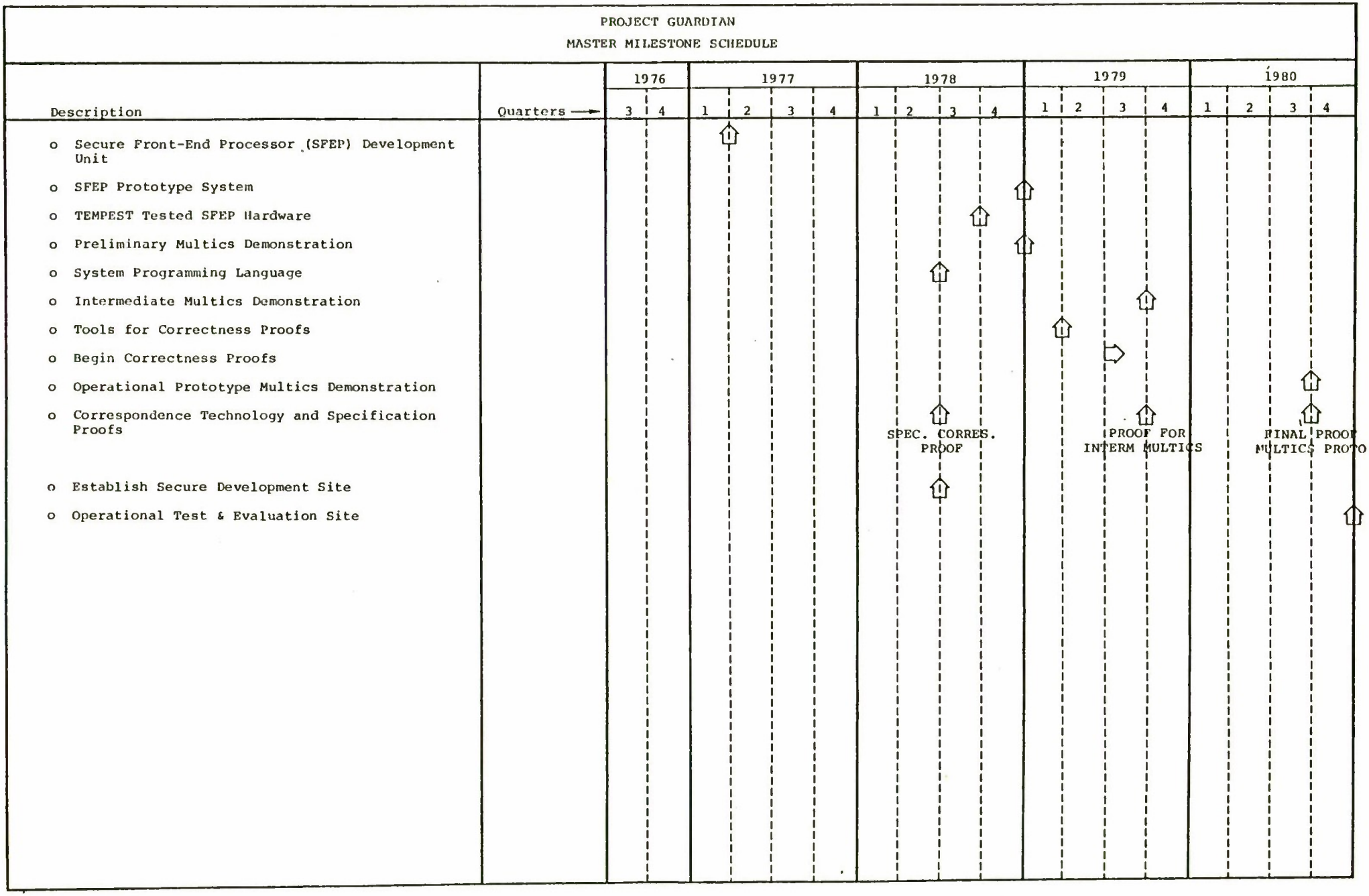
    Availability of a secure computer facility which provides a Top Secret controlled environment in which software development, system tests, and certification can be undertaken.

12. Operational Test and Evaluation Site

    Availability of an operational Multics site which will provide the environment for the Test and Evaluation of the secure Multics system under operational service conditions.

PROJECT GUARDIAN

MASTER MILESTONE SCHEDULE



To achieve these milestones, the plan described in this report has been broken into four major tasks. These tasks are:

Task 1 Multics Development

This task covers all activities for the simplification of the present Multics system, development of a security kernel for Multics, development of the supporting tools and techniques required to produce the kernel, support of the Secure Front-End Processor, development of technology and tools for certification, and beginning of the certification Process of the Multics kernel.

Task 2 Secure Front-End Processor Hardware

This task covers all activities for the specification, design, fabrication, testing, and design verification of the Secure Front-End Communications Processor.

Task 3 Secure Front-End Processor Software

This task covers all activities required for the specification, design, programming, development of technology and tools for certification, and beginning of the certification process of the kernel software for the Secure Front-End Communications Processor.

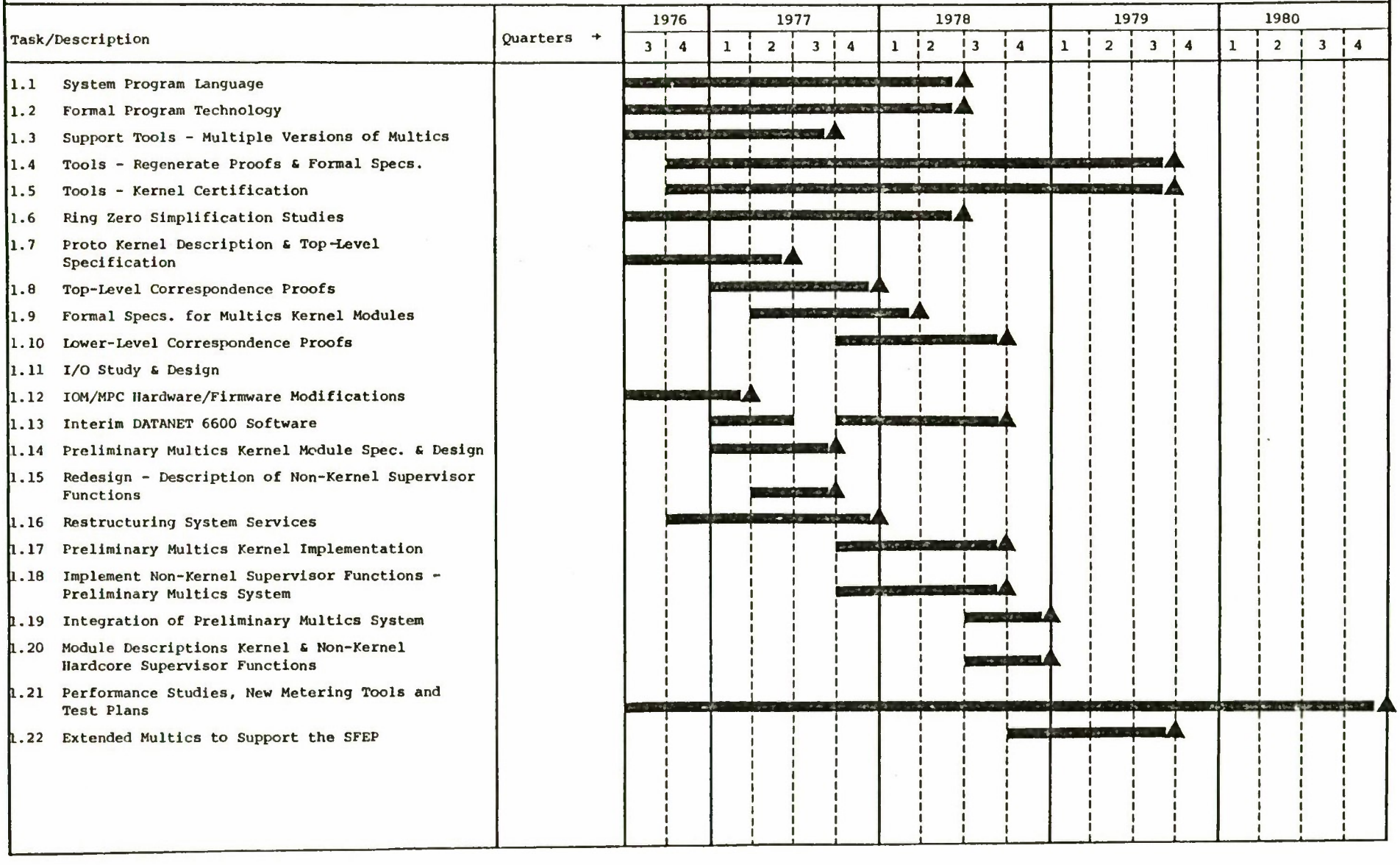
Task 4 Project Support

This task covers the activities required to administer the program, to support the task developers with documentation services, and to supply the computer services required for the completion of the three technical efforts above.

Each major task is the subject of a section of this report. The format of each section is an introduction to the major task followed by a definition and discussion of important subtasks. Each section concludes with a schedule which shows the resource requirements necessary to perform each task. The following schedules show the major subtasks of the program.

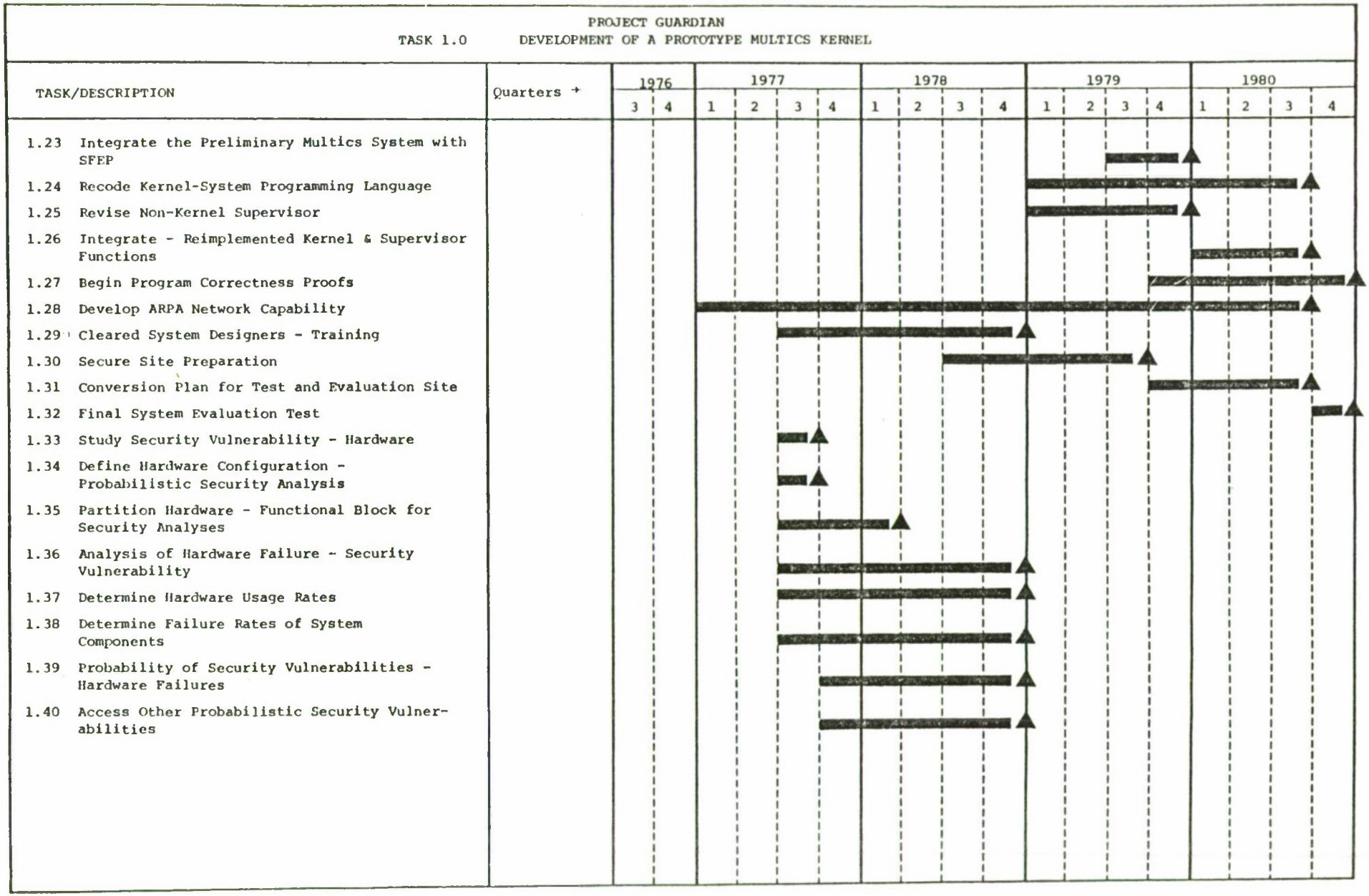
PROJECT GUARDIAN

TASK 1.0 DEVELOPMENT OF A PROTOTYPE MULTICS KERNEL



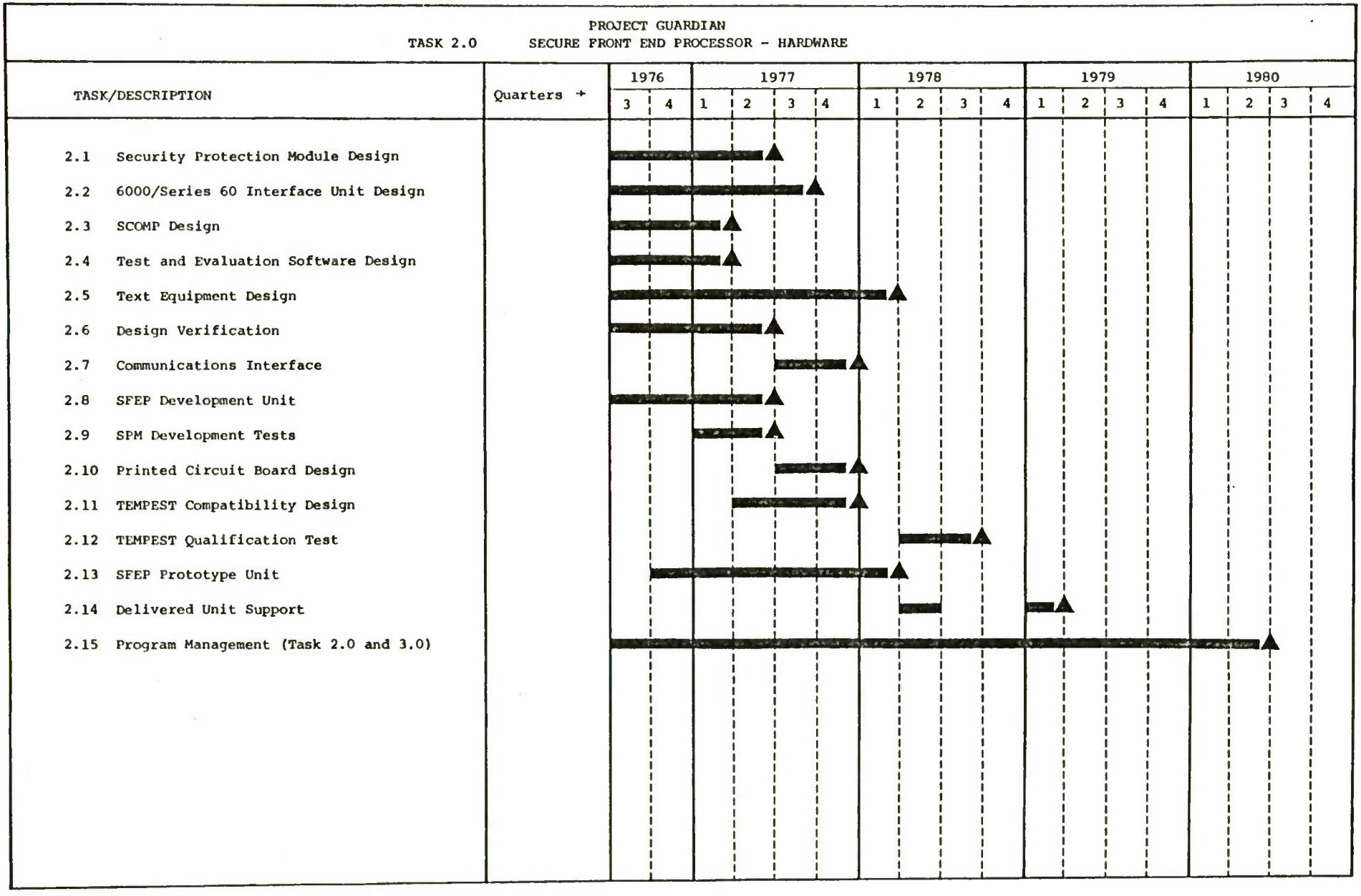
PROJECT GUARDIAN

TASK 1.0 DEVELOPMENT OF A PROTOTYPE MULTICS KERNEL



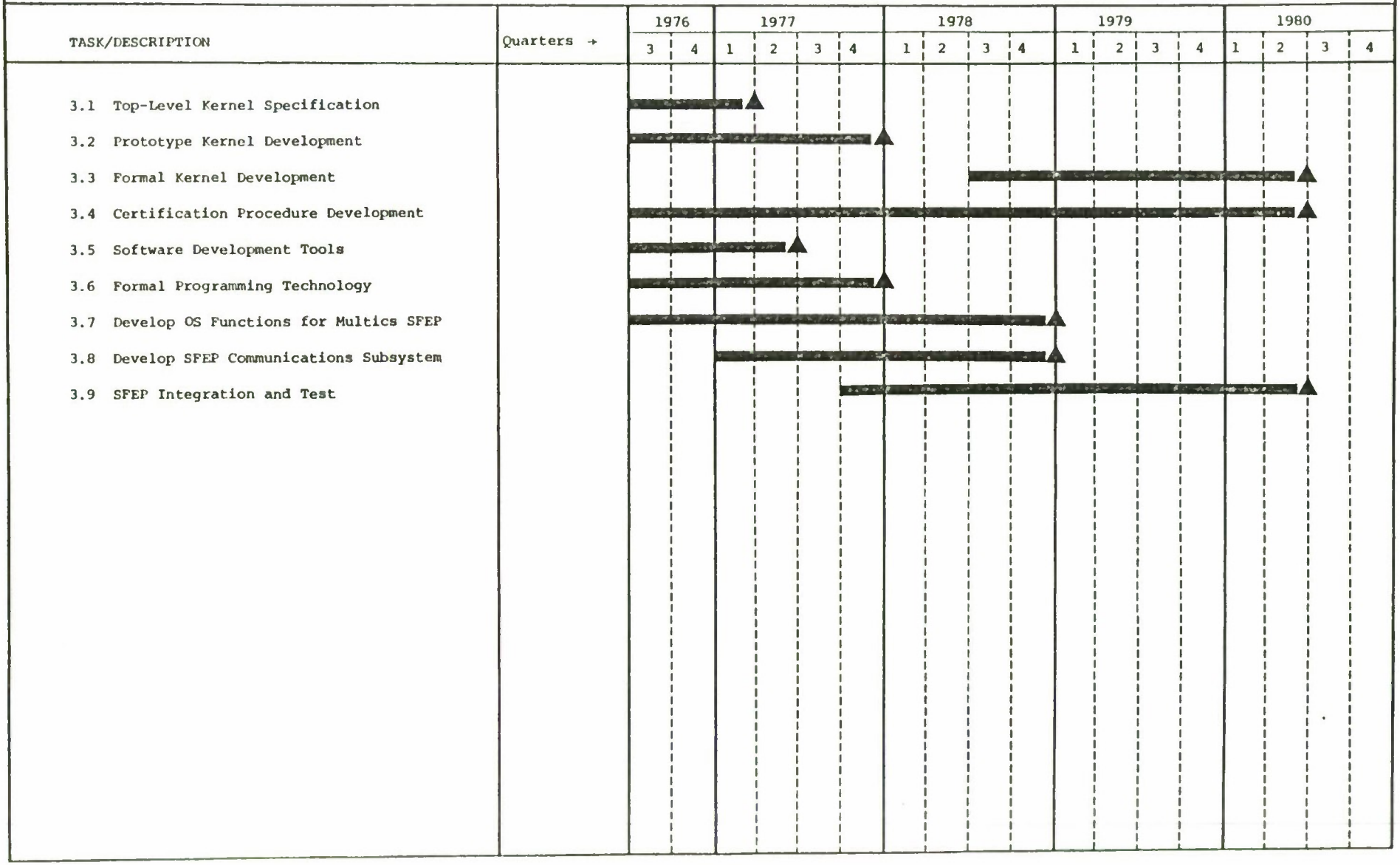
PROJECT GUARDIAN

TASK 2.0 SECURE FRONT END PROCESSOR - HARDWARE



PROJECT GUARDIAN

TASK 3.0 SECURE FRONT END PROCESSOR - SOFTWARE



PROJECT GUARDIAN
TASK 4.0 PROJECT CONTROL

| TASK/DESCRIPTION | Man Months Quarters → | 1976 | | 1977 | | | | 1978 | | | | 1979 | | | | 1980 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 4.1 Project Management | | | | | | | | | | | | | | | | | | | |
| 4.2 Configuration Management | | | | | | | | | | | | | | | | | | | |
| 4.3 Data Management/Documentation | | | | | | | | | | | | | | | | | | | |
| 4.4 Computer Resources* - Multics Service Site (GFE) | | | | | | | | | | | | | | | | | | | |
| 4.5 Computer Resources* - Multics Development Sites | | | | | | | | | | | | | | | | | | | |
| 4.5.1 Non-Secure Development Site | | | | | | | | | | | | | | | ▲ | | | | |
| 4.5.2 Secure Development Site | | | | | | | | | | | | | | | | | | | |
| 4.6 SFEP Resources | | | | | | | | | | | | | | | | | | | |

A set of supporting efforts has also been identified. These additional tasks will augment the system or improve its utility. These efforts are desirable additions to the project, but are not essential to meet the principle goals. The currently identified supporting efforts include:

A. Secure Data Base Management System

This effort covers the design, development, implementation, and test or certification of a Secure Data Base Management System to operate on the secure Multics. The system will allow multilevel access to a multilevel data base.

B. Cryptographic Multiplexor

This effort covers integration of a multiplexed cryptographic capability with the Secure Front-End Processor.

C. Secure Office Terminal Office Terminal

This effort includes development of procedures for use of crypto communications with the secure Multics, and procedures for use of a secure office terminal.

D. Security Audit and Surveillance

This effort covers the design and implementation of facilities for security audit and security surveillance of the secure Multics.

The supporting efforts are described in Appendix A of this report. The format of the sections on supporting efforts varies somewhat from that of the sections on major tasks, due to the large differences in the nature of the supporting efforts. There will probably be additional supporting efforts identified and considered during the progress of the major project.

## References


1. James P. Anderson, Computer Security Technology Planning Study, ESD-TR-73-51, October 1972.

2. R. Schell, P. Downey, G. Popek, Preliminary Notes on the Design of a Secure Military Computer System, MCI-73-1, January, 1972.

3. D. E. Bell, L. J. LaPadula, Secure Computer Systems, ESD-TR-73-278, The MITRE Corporation, Bedford, Mass.

4. W. R. Price, Implications of a Virtual Memory Mechanism for Implementing Protection in a Family of Operating Systems, PhD Thesis, Carnegie-Mellon University, June, 1973.

5. E. L. Burke, Synthesis of a Software Security System, MTP-154, The MITRE Corporation, Bedford, Mass.

6. D. E. Bell, E. L. Burke, A Software Validation Technique for Certification: The Method, ESD-TR-75-54, The MITRE Corporation, Bedford, Mass, December, 1973.

7. L. Robinson, P. G. Neumann, K. N. Levitt, A. Saxena, "On Attaining Reliable Software for a Secure Operating System", 1975 International Conference on Reliable Software, Los Angeles, Ca, April, 1975.

8. W. L. Schiller, Design of a Security Kernel for the PDP-11/45, The MITRE Corporation, Bedford, Mass, December, 1973.

9. W. L. Schiller, The Design and Specification of a Security Kernel for the PDP-11/45, The MITRE Corporation, Bedford, Mass, March, 1975.

10. L. Smith, Architectures for Secure Computing Systems, ESD-TR-75-51, The MITRE Corporation, Bedford, Mass, June, 1974.

11. J. C. Whitmore, A. Bensoussan, P. A. Green, A. M. Kobziar, J. A. Stern, Design for Multics Security Enhancements, ESD-TR-74-176, Honeywell Information Systems, Inc., 1974.

12. Access Isolation Mechanism Pre-Release Documentation, Honeywell Information Systems, Inc., McLean, Virginia, August, 1975.

13. W. L. Schiller, K. J. Biba, E. L. Burke, The Top Level Specification of a Multics Security Kernel, Working Paper WP-20377, The MITRE Corporation, Bedford, Mass., August, 1975.

# Section I

## Multics Development

### Introduction to the Development of a Prototype Kernel-Based Multics System

This part of the Integration Plan is concerned with modifications to the Multics system to demonstrate that the certifiable security kernel concept is feasible for a large-scale computer utility. Several technical reports will be prepared which together will describe the kernel-based Multics system.

The engineering approach for this effort is to develop three demonstration kernel-based Multics systems (evolving from Honeywell's standard product Multics system), each with increased function and correctness.

The first demonstration will show the feasibility of the kernel interfaces for Input/Output (I/O) and support of the nonkernel supervisor functions. Formal specifications for the kernel modules will be developed and correspondence proofs between these formal specifications and the security kernel model will be prepared. In developing this first demonstration, as much of the existing Multics code will be used as possible. All kernel development will be done in the PL/I and ALM languages. Correctness proofs of the kernel code will not be attempted. The current Multics DATANET 6600 software will be modified to support the communications interface to the kernel.

The second demonstration system will integrate the Secure Front-End Processor (SFEP) with the first demonstration system. The formal specifications will be revised as needed and the correspondence proofs will be regenerated.

The third demonstration system will incorporate the performance enhancements developed from the results of the first and second demonstration system performance studies. The security kernel will be recoded in the new system programming language which will be developed in parallel with the first demonstration system. Correctness proofs of the recoded kernel programs will be undertaken.

Informal technical reports describing the demonstration system effort as outlined under each of the following task descriptions will be available to the Air Force over the course of the project.

Task 1.1 System Programming Language Development

The programming language to be used in coding the third version of the prototype security kernel must be easy to understand if a correspondence between the formal program specifications and the executable code is to be made. The current system programming language, PL/I, does not meet this requirement as it exists today. Therefore, a programming language based on our experience with PL/I will be developed. This language will be a subset of the present PL/I language with some necessary modifications or extensions to facilitate certification. The language is also expected to be used during the SFEP kernel development effort of Task 3.5.

The specification for a programming language to be used in the implementation of the Multics security kernel will be developed. The major goals for the language will be:

a. easy to understand syntactic and semantic constructs

b. provide extensive support for manual validation techniques

c. fast efficient object code

d. flexible enough to support system programs

e. support Multics and SFEP kernel development efforts

A system programming language specification manual and an operable compiler for the language will be produced by this task. This specification must be available prior to the start of Task 1.24.

Stanford Research Institute (SRI) will act as a consultant during the language development on issues of support for correctness proofs.

Task 1.2 Establish A Formal Programming Technology

The technology for formal program specification is relatively new and has changed greatly during the last few years. The Multics environment will put some restrictions on the choice of a formal language which can be used effectively in the correspondence proofs. Therefore, some investigation in this area will be necessary.

The technology of formal program specification will be reviewed by Honeywell and SRI. A methodology will be adopted or refined which best suits the Multics and SFEP kernel-based systems and will be useful in the formal correspondence proofs for the kernels. The techniques for generating a formal correspondence proof between a mathematical model and a formal program

specification will be defined. These techniques will not utilize fully automated tools; instead, semiautomated tools to assist in manual proofs will be provided to aid in the tracking of the software throughout the life of the project.

A standards document will be written which describes the application of the formal specification method to Multics and the SFEP. Emphasis will be on approved practices for correct kernel program specification to support the correspondence proofs.

A detailed programming practices document will be developed which will build on the standards document. This work will be expanded as more is known about programming practices which will support correctness proofs. Particular attention will be paid to the application of the modified PL/I system programming language (developed under Task 1.1) to proving program correctness.

The format for documentation to best describe the structure of program modules and their interdependencies will be investigated. Techniques will be developed to ensure that the documentation will track changes to the modules properly. These techniques and software tools which implement them will be needed for the documentation effort described in Task 1.20. A technical manual describing the documentation techniques and tools will be prepared during this task.

Task 1.3 Develop Support Tools to Coordinate Multiple Versions of Multics

After the completion of the first demonstration system, there will be two distinct Multics systems supporting (nearly) the same user interface (i.e., the standard product Multics and the kernel-based demonstration system Multics.) These two systems must track all changes to the Multics user interface as closely as possible. Software tools to aid in the comparison of library programs for the two systems are needed. Currently available library maintenance software tools will be modified and extended to support the second collection of programs dedicated to this development effort. These maintenance tools will identify any differences between the two Multics systems on a continuing basis.

This task will involve the development of software tools (and possibly manual techniques) to support the maintenance of parallel Multics systems. As a minimum, complete library comparison programs are needed to identify modules which have been updated in one system and not in the other.

A technical manual describing the support tools will be prepared during this task.

Task 1.4 Develop Tools to Generate and Regenerate Correspondence Proofs and Formal Specifications

During the course of the project, there will be several changes to the kernel module specifications as design problems are resolved. For each change, the formal specifications and correspondence proofs may have to be updated.

This task covers, in part, the SRI effort of designing semiautomated software tools for generating and modifying the formal specifications and for generating and regenerating correspondence proofs. These tools presently exist and have been demonstrated for other applications by SRI. They must be transferred to Multics to be applicable to the Multics and SFEP specification efforts.

Available tools developed under this task will be used during the development of the second demonstration system. They will also be applied during the design of performance enhancements for the third demonstration system. These tools will also be applied to the correspondence proving task for the SFEP kernel activity in Task 3.4.

Two progress reports will be prepared during the course of this task. At the conclusion, a technical manual describing the tools will be prepared.

Task 1.5 Develop Tools and Techniques for Kernel Certification and Recertification

The long-range goal of this project is to show that the kernel-based Multics system can be certified as secure. To this end, a detailed specification of the criteria for certification must be developed. This specification must describe the certification process from the formal model to the executable software and the hardware. The certification process must lend itself to an orderly recertification as the kernel evolves and as hardware changes are incorporated.

This task covers, in part, the SRI effort of defining the criteria for certification of the kernel-based Multics and SFEP systems and in designing tools and techniques for initial certification and recertification as the system changes. Some research into the handling of concurrency and exception conditions will be required. The development of these tools will be at the option of the Air Force.

Task 1.6 Ring Zero Simplification Studies

This task covers the MIT Laboratory for Computer Science (formerly known as Project MAC) efforts in reducing the size of ring zero. Many of the areas under current investigation show promise for generating design approaches which would be useful in the kernel development.

The results of these simplification studies will be reviewed in detail as the kernel design progresses. The ideas described will be factored into the kernel design as appropriate.

Each area of study will be described in technical memos as appropriate during the course of this task.

Task 1.7 Develop An Informal Prototype Kernel Description and A Top-Level Formal Prototype Kernel Specification

This task involves expanding the functional description of the prototype kernel into a more detailed specification of modules that will form the kernel, including their interactions. The security of the initial state of the kernel and all state transitions will be shown.

After the initial work of this task has been completed, a design review with the Air Force will be held to ensure that the Air Force and Honeywell are in agreement that the kernel description will support the security model as well as the requirements of engineering feasibility.

The kernel description will now be the basis for the development of top-level formal prototype kernel specifications. Each of the kernel interfaces will be described using the formal program specification methodology adopted for the Multics system under Task 1.2.

The informal kernel description and the top-level formal kernel specification will be integrated into a detailed prototype kernel specification.

Task 1.8 Develop Top-Level Correspondence Proofs

During this task, Honeywell and SRI will work together to prove that the top-level formal prototype specification corresponds to the Air Force security model. Progress on this task assumes that the informal prototype kernel description provided to the Air Force at the design review from Task 1.7 has been accepted. This task will also serve as the basis for the similar SFEP Task 3.3 which is described later.

A technical report describing the top-level correspondence proofs will be prepared at the conclusion of this task.

Task 1.9 Develop Complete Formal Specifications for the Multics Kernel Modules

This task will extend the top-level formal prototype kernel specification to the lower levels to include the various modules within the security kernel. This will include the full specification of the interfaces between all modules and the functions that are performed by each module.

An update to the detailed prototype kernel specification will be prepared at the conclusion of this task.

Task 1.10 Develop Lower-Level Correspondence Proofs of the Multics Kernel

During this task, Honeywell and SRI will work together to prove that the complete formal specifications of the prototype kernel modules follow the top-level kernel specifications and thereby follow the security model prepared by the Air Force. A complete report describing the correspondence between the model and the top-level kernel specification and between the top-level and lower-levels will be prepared at the completion of this task.

Task 1.11 I/O Study and Design

This task will determine how external I/O should be handled in the kernel-based Multics system. All internal I/O will be handled by the kernel through an I/O Multiplexer (IOM). All external communications will be handled through an SFEP. Between these two extremes an engineering evaluation will be made to determine how other forms of external I/O will be handled within the system. Such factors as physical constraints, data transmission speeds, system initialization, system crash recovery and efficiency must be considered along with the security model to determine for each I/O device what system architecture best satisfies all the constraints.

Early in the course of this task, a design review will be held with the Air Force to present all the factors on handling external I/O within the kernel-based Multics system. Complete agreement between Honeywell and the Air Force on the design approach is necessary before work can continue on the project.

A technical report describing the recommended approach to handling external I/O in the kernel-based system will be prepared during this task.

Task 1.12 IOM/MPC Firmware/Hardware Modifications

It is recognized that the I/O study may recommend certain modifications to the IOM in the form of hardware changes or to the Microprogrammed Controllers (MPC) either in the form of hardware or firmware changes. This task will outline such modifications in detail and oversee their correct implementation.

All changes to hardware and/or firmware will be described in the I/O study report. Hence, no additional reports will be generated during this task.

Task 1.13 Develop Interim DATANET 6600 Software to Support SFEP Interface to the Kernel

Testing of various implementation phases of the security kernel must proceed before the planned completion of the SFEP. Therefore, the existing Multics DATANET 6600 software may be modified to support the interfaces to the Multics security kernel, as defined under Task 1.11, so that the functions of the security kernel can be confirmed. Formal specifications for these modifications and correspondence proofs will not be prepared since this is an interim solution to handling the SFEP interface to the security kernel.

The prime purpose of this task is to decouple the DATANET 6600 communications interface from the Multics kernel and SFEP kernel development efforts for as long as possible. This approach minimizes potentially undesirable dependencies between the Multics and SFEP development efforts. The intent of this task is not to recode the DATANET 6600 software but, rather to develop an interim interface to the SFEP kernel by modifying Multics communications software and/or the DATANET 6600 software to support the required interim interface to the SFEP kernel. These modifications will be those necessary to demonstrate the kernel/non-kernel interface on the 6180 processor as well as simulate the kernel to kernel interface to the SFEP.

Design memos describing the modifications to the Multics DATANET 6600 software will be prepared during this task. These modifications will be limited to those necessary to demonstrate the kernel/non-kernel interface on the 6180 processor.

Task 1.14 Preliminary Multics Kernel Module Specifications and Design

This task will take the formal specifications of the lower-levels of the security kernel and convert them into calling sequences and argument descriptions for each module in preparation for the first implementation.

Task 1.15 Redesign and Informal Description of Non-Kernel Supervisor Functions

This task involves modifying all supervisor interfaces that were affected by the kernel design. Good engineering practice requires many of these interfaces to be protected from tampering by the user for denial of service considerations. The task also involves the mapping of the current Multics user interface into new supervisor functions or into direct kernel interfaces.

An informal description of the nonkernel supervisor design revisions will be prepared during this task.

Task 1.16 Restructuring System Services

The current Multics system provides some system support functions which are logical extensions of the ring zero kernel. They are implemented as separate processes which are trusted to support the security model rules for access, but must be able to operate outside the security access constraints of the kernel. Some examples of these are: Answering Service, backup and retrieval, and system I/O.

This task will examine these trusted processes to find alternative ways to provide their functions with less privilege. The IPC and message facilities which allow users of all security levels to communicate with each other and with the trusted processes will be examined for inclusion in the kernel design. The device and demountable media resource control mechanisms will be restructured to account for the access rules of the model.

Honeywell will prepare informal design memos describing each of the system services that are restructured as part of this task.

Task 1.17 Preliminary Multics Kernel Implementation

This task covers the coding and testing of the kernel for the first demonstration system. Each of the kernel modules defined under Tasks 1.7, 1.9, and 1.14 will be coded in the PL/I or ALM languages. Many of the modules may be variations of existing ring zero modules. As much of the existing code will be used during this implementation as possible.

Task 1.18 Implementation of the Non-Kernel Supervisor Functions for the Preliminary Multics System

This task runs in parallel with the first prototype kernel implementation and includes the coding and testing of those nonkernel supervisor modules which were defined in Tasks 1.15 and 1.16. All coding will be done in the PL/I or ALM languages and

much of the existing Multics supervisor code will be used.

Task 1.19 Integration of the Preliminary Multics System

This task is the first attempt to bring together the kernel and nonkernel supervisor functions and the interim DATANET 6600 software to support the user interface of the current Multics system. At the completion of this task, the first Multics system based on a security kernel will be demonstrated.

Task 1.20 Module Descriptions for the Kernel and Non-Kernel Hardcore Supervisor Functions

Module descriptions for all hardcore supervisor programs will be prepared using the tools and techniques developed under Task 1.2 for correctly documenting program modules. These descriptions will detail the final program calling sequences and argument descriptions.

The module descriptions together with the formal kernel specification and the nonkernel supervisor revisions will form the complete documentation of the kernel-based hardcore supervisor.

Task 1.21 Performance Studies, New Metering Tools, Test Plans and Reports

This task is responsible for estimating, measuring and reporting on the performance, compatibility and security of the kernel-based Multics system through all three demonstration versions.

Useful measures of performance, compatibility and security will be defined. Estimates of goals to be attained in each of these areas will be made. Measurements or estimates of progress toward these goals will be prepared for inclusion in periodic status reports.

This task will define new metering tools which can be included within the kernel and the nonkernel supervisor to determine how the system is performing after its initial implementation and system integration. It will involve detailed examination of the design decisions in the development of the modules within the kernel and within the nonkernel supervisor to determine their impact on overall system performance. The results will be used during the recoding of the kernel for the third demonstration (see Task 1.24). A report describing the performance studies, metering tools, and estimates on performance of the kernel-based system will be prepared prior to the first demonstration.

Plans for each of the three system demonstration tests will be prepared for Air Force review and approval. These plans will outline the procedures to be followed during the test, the performance measurements to be made, and how the system will demonstrate the expected degree of compatibility. The results of each of the three demonstration tests will be provided in a report within 60 days following the test.

After completing the third demonstration test, a comprehensive report covering the entire project will be prepared. This report will describe: design goals of the project, system architectural design decisions, evolution of the kernel-based Multics, test results, system performance, compatibility between the operational prototype secure Multics and the standard Multics systems, the criteria for certification, and how the kernel-based Multics system meets the certification criteria.

Task 1.22 Extend Multics to Support the SFEP

When the detailed design of the SFEP is completed, final work on the Multics support for the SFEP can begin. Also, the kernel interface to the SFEP can be refined from the original DATANET 6600 interim design to the final SFEP design. This will be done in preparation for the second demonstration. Also, see Task 1.13 above which describes this interim interface.

This task covers the design and implementation of changes to the kernel, the nonkernel supervisor, the administrative functions and initialization functions to support the SFEP. The formal specifications of the kernel modules and the correspondence proofs will be updated as necessary. The hardcore supervisor module descriptions will be updated to include the complete SFEP interface.

Task 1.23 Integrate the Preliminary Multics System with the SFEP

This task covers the system tests to checkout the SFEP on the kernel-based Multics system. At the completion of this task, the second system will be ready for demonstration.

Task 1.24 Recode the Kernel in the System Programming Language and Integrate Performance Improvements

The demonstration of the first system will show the feasibility of the kernel interfaces and the restructuring of the Multics supervisor. The next major step toward certification will be to show the correspondence between the formal module descriptions and the implementation language.

To accomplish this, the kernel modules must be recoded in the system programming language developed under Task 1.1. All recoding will be done at the secure Multics installation (see Task 1.30) by a special group of system designers who are cleared to Top Secret. All formal specifications, module descriptions, program listings and test data from the first and second demonstration systems will be available to the system designers at the secure site. Any program listings, specifications or machine readable information (e.g., system tapes, card decks or dump tapes) prepared at the secure site, other than the protected copies, may be freely removed and will be unclassified.

While the kernel modules are being recoded, they will be examined for performance improvements based on the study results in Task 1.21. These performance improvements will be incorporated where applicable.

The formal kernel specifications and correspondence proofs will be updated as needed. The kernel module descriptions will be revised.

Task 1.25 Revise the Non-Kernel Supervisor for Performance Improvements

The performance studies described in Task 1.21 will show some areas outside the kernel which adversely affect performance. These areas will be redesigned for performance improvements in parallel with the recoding of the kernel.

This task covers the redesign to implement these performance enhancements. The revisions for the nonkernel supervisor will be updated and the module descriptions will be revised as needed.

Task 1.26 Integrate the Re-implemented Kernel and Supervisor Functions

The revisions to the kernel made under Tasks 1.24 and 1.25 will be integrated into the complete operational prototype secure Multics system to form the basis for the third demonstration.

Initial integration testing will be done at the Honeywell non-secure development site, using a copy of the kernel produced at the secure development site (see Task 1.30). All changes to the kernel will be made by cleared system designers at the secure site to maintain the integrity of the kernel. Final development testing and the demonstration test will be performed at the secure site using real classified data and a group of cleared users requiring classified data processing.

This task covers the integration and system tests to demonstrate the kernel-based Multics system that is to be certified. The

results of this effort will be the third demonstration system.

Task 1.27 Program Correctness Proofs

After both the SFEP and Multics kernels have been recoded in the appropriate system programming language, the correctness proof techniques developed under Task 1.5 and Task 3.4 will be applied. Correctness proofs and correspondence to the formal specifications for kernel program modules will be accomplished at the source code level only. Correctness proofs of program object code is beyond the scope of this effort.

This task initiates the process of showing that the security kernels meet the certification criteria. The magnitude of this task is highly dependent on how far the state of the art of program correctness proofs has been advanced during the course of the project.

A technical report describing the work on final certification will be prepared.

A review of the progress on this task will be held about three (3) months after its start.

Task 1.28 Develop an ARPA Network Capability for the Operational Prototype Multics System

Multics development sites currently provide the capability to use the ARPA network with a special hardware and software interface to an ARPA network Interface Message Processor (IMP) computer. At the present time, this capability is not part of the standard Multics product.

This task will utilize the technology and software/hardware design of the current ARPA network interface to develop an ARPA network capability for the secure Multics demonstration system. The current host to IMP protocol will be used. The actual network connection will be through the SFEP. The ARPA network is not physically secure and does not provide logical identification of security classification for messages. Therefore, a multilevel security interface cannot be provided. However, the design will not preclude future multilevel operation from being implemented if a secure interface for the ARPA network is eventually defined. Any changes to the ARPA network protocols are beyond the scope of this task. An implementation plan and design specification will be prepared. The ARPA network software (and possibly hardware) designed under this task will be integrated into the second or third demonstration Multics system as defined in the implementation plan. All coding will be done by cleared personnel since the network support capability must be integrated late in the course of the project. Final testing will be done

using the ARPA network connection at the secure site (see task 1.30).

Task 1.29 Train Cleared System Designers

The third demonstration kernel software will be developed by system designers with Top Secret security clearances. Assuming clearable trained system designers would not be available with replacement, this task provides for the training of four system designers who have the necessary security clearance. They will work with the uncleared system designers for 18 months before beginning the recoding of the kernel for the third demonstration system.

Task 1.30 Secure Site Preparation

A secure Multics system installation controlled at a security level of Top Secret is needed in recoding the security kernel for the third demonstration system. Only system designers cleared to Top Secret can be allowed to recode the security kernel, if the integrity of the implementation is to be certified to this level.

This task covers the preparation of: the physical site, the hardware, the communications systems, the operating procedures, and the site support procedures. This task also provides specialized training for site support personnel in the operation and maintenance of the kernel-based Multics.

The secure site will serve as: a secure repository of the security kernel; a secure location to edit and compile programs which comprise the kernel; a tool for generating system tapes containing the secure kernel; a final development test site for checkout of interim kernel software; a service site for processing Top Secret data for a group of cleared users; and a interim test and evaluation site for the third demonstration system using the group of cleared users as a base for system load. Operating time will be divided among these activities when they are mutually exclusive.

The specific elements of this task are:

a. Work with the Air Force to establish a group of users who need to do Top Secret processing. (one year in advance of the third demonstration system.)

b. Work with the Air Force in choosing a secure site location.

c. Consult on the preparation of the secure site for hardware installation.

d. Procure the Honeywell Series 60 Level 68 (or equivalent) hardware

e. Work with the Air Force to procure an ARPA network IMP and communications interface.

f. Define site operations and security procedures.

g. Train Top Secret cleared Air Force operators and communications specialists.

h. Train Top Secret cleared Honeywell Site Analysts and Field Engineers.

Task 1.31 Conversion Plan for a Test and Evaluation Site

This task will develop a detailed plan to transfom a site (e.g., AFDSC) from an operational installation, which is offering the Multics standard system as a service, to a site which will provide a kernel-based Multics system for test and evaluation. The application of this conversion plan will be as follows:

a. Implement the conversion plan and evaluate the conversion aids at the secure development site.

b. Revise the secure Multics capability at the secure development site as necessary and modify the Conversion Plan as appropriate for Air Force concurrence.

c. Implement the plan at any other Multics site which is to be used for a test and evaluation of the kernel-based system.

This plan will first review current operational procedures and then identify all additional hardware, specialized software to be implemented and exercised (e.g., storage system conversions, etc.), additional documentation, and required training for the changeover of a Multics site so that the site can use the kernel-based Multics system. The problem of reverting a site back to support of standard Multics hardware and software will also be addressed.

The primary output of this task is a detailed Conversion Plan.

Task 1.32 Final System Evaluation Test Period

After the successful completion of the third demonstration, the kernel-based Multics system will remain operational at the secure Multics installation. The system will be run in multilevel security mode with users chosen by the Air Force. Uncleared users will be allowed to use the system only if approved by the

appropriate Air Force authority. Classified data processing will be done on a regularly scheduled basis as a Multics service site.

Task 1.33 Study Security Vulnerability due to Hardware Failure

The purpose of this task is to identify hardware events that result in a security vulnerability in the secure Multics system. For this task, it will be assumed that an event either does or does not result in a security vulnerability. That is, all security vulnerabilities are considered to be equally serious. A separate task may be established later to assign a "degree of seriousness" rating to each security vulnerability event.

The previous work along these lines for the Secure Communications Processor (SCOMP) will be examined and the methodology will be used where practical.

The output of this task will be a report which identifies the security vulnerability events and discusses the criteria and methodology used in this task.

Task 1.34 Define Hardware Configuration for Probabilistic Security Analysis

The purpose of this task is to identify equipment for which failure rates must be determined and to provide hardware configuration inputs to the model that will calculate the system's probabilistic vulnerability rating due to hardware failures. Subtasks include:

a. Identifying the equipment that will be in the Project Guardian Multics system for this analysis.

b. Identifying the changes that will be made to existing equipment.

c. Determining the major characteristics of the new equipment. This includes the gross implementation technology and approach.

d. Deciding on the quantities and sizes of the equipment that will comprise a "typical" or base system.

The scope of the equipment covered by this task ranges from the processor to peripheral devices and the SCOMP. The output of this task will be a report that details the expected composition of the Multics system to be used in the verification tasks below (Task 1.35-Task 1.40).

Task 1.35 Partition the Hardware into Functional Blocks for the Security Analysis

In order to evaluate the impact of certain failures in the "security sensitive" portions of the hardware, it is necessary to delineate those portions of the hardware. That is the purpose of this task. The result can be viewed as a detailed block diagram of the security sensitive hardware in a system, although the actual results may be in a tabular form, and not necessarily in conventional block diagrams.

The challenges of this task are to establish a security perimeter and to subdivide the portion of the system within that security perimeter into "suitably sized" blocks. Their functional size must be appropriate for meaningful use in other tasks related to the probabilistic measure of security vulnerabilities due to hardware failure. Therefore, part of this task involves developing criteria to use in establishing a security perimeter and in subdividing a large computer system into the "appropriate" blocks.

This task effort will include examining the applicability of reliability studies and Failure Modes and Effects Analysis (FMEA) studies that have been or are being done on Honeywell equipment used in Series 60/6000 systems. Where practical, the methodologies and/or results of such studies will be applied, extended, or adapted to this task's needs.

Task 1.36 Analysis of Hardware Failure as A Security Vulnerability

This task involves deciding whether or not a security vulnerability exists for each failure mode of each functional block of the hardware.

The speed and thoroughness with which hardware failures are detected is an important aspect of deciding whether or not a failure gives rise to a security vulnerability. The impact of incomplete failure detection may be lessened by periodically running tests which verify hardware functions that may be deficient in automatic failure detection. These considerations will be factored into the security vulnerability assignments. Some recommendations may be made for hardware changes and/or the running of periodic "health checks" to be certain that internal hardware failure detection is completely operational. The output of this task will be a set of data in a database that is used by a later task.

This task includes an assessment of the probable effects of transient or intermittent hardware failures, as well as solid failures. The results of this task may suggest recommendations as to hardware functions that should be avoided by the security

kernel. These will be factored into programming standards and language tasks.

Task 1.37 Determine Hardware Usage Rates

The security vulnerability of a portion of the hardware is directly proportional to the degree to which that portion of the hardware is utilized. This task determines the usage rates for those functional hardware blocks whose failure may give rise to a security violation.

The data will be structured to reflect the variable range of the usage by different portions of the software, firmware, and the generic actions by user programs. The output from this task will be a set of data in a database that is used by a later task.

Accomplishing this task will require extensive hardware measurements on an operational system plus substantial analysis of how the software uses the hardware.

However, the output of this task will be useful for much more than analyzing security vulnerabilities. Analysis of the data gathered during this task will disclose ways to improve system performance by different utilization of the existing hardware. Ways to change the hardware, or improvements to make in the next generation of hardware, may also become apparent.

Task 1.38 Determine Failure Rates of System Components

Much raw data already exists for the equipment used in Multics systems (i.e., Series 60/6000). This task involves collecting existing data, identifying portions of the Multics system not covered by the existing data, and obtaining suitable data for those non-covered portions of the system. The data will be organized in a database to correspond to the functional model of the hardware that was established in an earlier task for purposes of evaluating the security vulnerabilities. (See Task 1.36). That is, data for individual piece parts (chips) will be combined to yield data for a low-level function of the hardware (e.g., an adder).

The data will also be structured to reflect the various failure modes that are of importance for potential security vulnerabilities. This data base will be refined as necessary during the life of the project. Such refinements are expected to be minimal.

Task 1.39 Probability of Security Vulnerabilities due to Hardware Failures.

This task combines the numerical results of the previous tasks to calculate the probability of security vulnerabilities due to hardware failures. This task will:

a. Develop computer programs.

b. Establish data bases.

c. Exercise the programs.

d. Collate the results

The output of this task will be a report that summarizes the calculation methodology, summarizes the content of the data bases, and presents the numerical results of performing the calculations.

Task 1.40 Assess Other Probabilistic Security Vulnerabilities

This task involves identifying situations or events other than hardware failures which are of a probabilistic nature and can lead to security violations. The impact of such events and situations will then be evaluated. The only candidate identified so far for this task is falsified authentication of users (password guessing).

The output from this task will be a report that identifies the situations and the security vulnerability probabilities.

Resource Requirements

The following table indicates the resource requirements necessary for the subtasks of Task 1.0.

MANPOWER
MANMONTHS/QUARTER

TASK 1.0

PROJECT GUARDIAN
DEVELOPMENT OF A PROTOTYPE MULTICS KERNEL

| Task/Description | Quarters → | 1976 | | 1977 | | | | 1978 | | | | 1979 | | | | 1980 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1.1 System Program Language | | 3.0 | 10.5 | 7.5 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | | | | | | | | | | |
| 1.2 Formal Program Technology | | 9.0 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 3.0 | 3.0 | 3.0 | 3.0 | | | | | | | | |
| 1.3 Support Tools - Multiple Versions of Multics | | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | | | | | | | | | | | | | |
| 1.4 Tools - Regenerate Proofs & Formal Specs. | | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 6.0 | 6.0 | 6.0 | 6.0 | | | | | | | |
| 1.5 Tools - Kernel Certification | | | 1.5 | 1.5 | 1.5 | 1.5 | 3.0 | 3.0 | 12.0 | 12.0 | 12.0 | 12.0 | 6.0 | 6.0 | | | | | |
| 1.6 Ring Zero Simplification Studies | | 18.0 | 18.0 | 14.0 | 14.0 | 14.0 | 14.0 | 5.0 | 5.0 | | | | | | | | | | |
| 1.7 Proto Kernel Description & Top-Level Specification | | 10.5 | 10.5 | 10.5 | 3.0 | | | | | | | | | | | | | | |
| 1.8 Top-Level Correspondence Proofs | | | | 3.0 | 6.0 | 6.0 | 3.0 | 1.5 | | | | | | | | | | | |
| 1.9 Formal Specs. for Multics Kernel Modules | | | | | 3.0 | 6.0 | 6.0 | 6.0 | | | | | | | | | | | |
| 1.10 Lower-Level Correspondence Proofs | | | | | | | 1.5 | 6.0 | 6.0 | 3.0 | 3.0 | | | | | | | | |
| 1.11 I/O Study and Design | | - | - | | | | | | | | | | | | | | | | |
| 1.12 ICM/MPC Hardware/Firmware Modifications | | 1.5 | 1.5 | 1.5 | | | | | | | | | | | | | | | |
| 1.13 Interim DATANET 6600 Software | | | | 3.0 | 3.0 | | 3.0 | 3.0 | 3.0 | 3.0 | | | | | | | | | |
| 1.14 Preliminary Multics Kernel Module Spec. and Design | | | | 6.0 | 6.0 | 6.0 | | | | | | | | | | | | | |
| 1.15 Redesign - Description of Non-Kernel Supervisor Functions | | | | | 6.0 | 6.0 | | | | | | | | | | | | | |
| 1.16 Restructuring System Services | | | 3.0 | 6.0 | 6.0 | 6.0 | 3.0 | | | | | | | | | | | | |
| 1.17 Preliminary Multics Kernel Implementation | | | | | | | 9.0 | 9.0 | 6.0 | 3.0 | | | | | | | | | |
| 1.18 Implement Non-Kernel Supervisor Functions - Preliminary Multics System | | | | | | | 12.0 | 15.0 | 12.0 | 9.0 | | | | | | | | | |
| 1.19 Integration of Preliminary Multics System | | | | | | | | | | 12.0 | 15.0 | | | | | | | | |
| 1.20 Module Descriptions Kernel & Non-Kernel Hardcore Supervisor Functions | | | | | | | | | | 3.0 | 3.0 | | | | | | | | |
| 1.21 Performance Studies, New Metering Tools & Test Plans | | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 9.0 | 9.0 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 | 6.0 |
| 1.22 Extended Multics to Support the SFEP | | | | | | | | | | | 3.0 | 6.0 | 3.0 | 3.0 | | | | | |

MANPOWER
MANMONTHS/QUARTERS

PROJECT GUARDIAN
TASK 1.0 DEVELOPMENT OF A PROTOTYPE MULTICS KERNEL

| TASK/DESCRIPTION | Quarters → | 1976 | | 1977 | | | | 1978 | | | | 1979 | | | | 1980 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1.23 Integrate the Preliminary Multics System with SFEP | | | | | | | | | | | | | | 3.0 | 3.0 | | | | |
| 1.24 Recode Kernel-System Programming Language | | | | | | | | | | | | 15.0 | 15.0 | 15.0 | 12.0 | 9.0 | 6.0 | 6.0 | |
| 1.25 Revise Non-Kernel Supervisor | | | | | | | | | | | | 9.0 | 9.0 | 9.0 | 9.0 | | | | |
| 1.26 Integrate - Reimplemented Kernel & Supervisor Functions | | | | | | | | | | | | | | | | 9.0 | 12.0 | 12.0 | |
| 1.27 Program Correctness Proofs | | | | | | | | | | | | | | | 7.5 | 10.5 | 10.5 | 9.0 | 9.0 |
| 1.28 Develop ARPA Network Capability | | | | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | |
| 1.29 Cleared System Designers - Training | | | | | | 12.0 | 12.0 | 12.0 | 12.0 | 12.0 | 12.0 | | | | | | | | |
| 1.30 Secure Site Preparation | | | | | | | | | | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | | | | | |
| 1.31 Conversion Plan for Test & Evaluation Site | | | | | | | | | | | | | | | 1.5 | 1.5 | 1.5 | 3.0 | |
| 1.32 Final System Evaluation Test | | | | | | | | | | | | | | | | | | | 3.0 |
| 1.33 Study Security Vulnerability - Hardware | | | | | | 2.0 | | | | | | | | | | | | | |
| 1.34 Define Hardware Configuration - Probabilistic Security Analysis | | | | | | 3.0 | | | | | | | | | | | | | |
| 1.35 Partition Hardware - Functional Block for Security Analyses | | | | | | 4.0 | 8.0 | 4.0 | | | | | | | | | | | |
| 1.36 Analysis of Hardware Failure - Security Vulnerability | | | | | | 4.0 | 6.0 | 9.0 | 6.0 | 3.0 | 3.0 | | | | | | | | |
| 1.37 Determine Hardware Usage Rates | | | | | | 3.0 | 9.0 | 12.0 | 12.0 | 6.0 | 6.0 | | | | | | | | |
| 1.38 Determine Failure Rates of System Components | | | | | | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | | | | | | | | |
| 1.39 Probability of Security Vulnerabilities - Hardware Failures | | | | | | | 3.0 | 3.0 | 3.0 | 3.0 | 6.0 | | | | | | | | |
| 1.40 Access Other Probabilistic Security Vulnerabilities | | | | | | | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | | | | | | | | |

# Section II

## Secure Front-End Processor Hardware

### Introduction and Overview

This task produces the hardware base for a Secure Front-End Processor (SFEP). The basic technical approach for the SFEP hardware design is to expand the selected minicomputer now in development within Honeywell for applications to include an essentially autonomous Security Protection Module (SPM). The extended minicomputer can be used in a stand-alone environment and is then known as a Secure Communications Processor (SCOMP). The SPM, when integrated into the minicomputer, will provide the hardware controls necessary for a Secure Communications Processor. When an interface unit for a host 6000/Series 60 computer, a TEMPEST unit, and selected peripherals are added, the required Secure Front-End Processor results.

Secure Front-End Processors are required for large-scale systems to extend the security perimeter to include external communications and input/output facilities. This part of the Integration Plan encompasses the design, development, and fabrication of two Secure Front-End Processors for eventual integration into hardware configurations for hardware testing, SFEP software development, and an operational prototype secure Multics system.

This effort will complete the design, fabrication, integration, and test evaluation of SFEP hardware prototypes. Specific tasks are:

a. Design, development and fabrication of Security Protection Modules and 6000/Series 60 Interface Units to permit the integration of modified minicomputers into large scale systems to function as Secure Front-End Processors (SFEP).

b. The design and fabrication of modules and components required for TEMPEST compatibility.

c. The fabrication of a non-ruggedized developmental SFEP unit to serve as a hardware and software checkout, development, and demonstration facility.

d. The fabrication of a ruggedized prototype SFEP unit for TEMPEST qualification testing and Multics integration tests.

Task 2.1 Security Protection Module (SPM) Design

The SPM is an address translation unit necessary to support the access control requirements of a security kernel. It is an autonomous module which can be integrated into the minicomputer to provide segmented addressing on a per process basis.

The SPM contains a fast-access cache memory for storage of data descriptors and related access privileges, provisions for multiple machine states (rings of protection), storage for access and call brackets, fault directing vectors and related controls necessary to support the Reference Monitor concept.

The SPM design task includes the following subtasks: detailed logical design, mechanical design, documentation, parts selection, development and product specifications, and support to design verification and SPM evaluation tests. The output from this task will be the design documentation which will permit fabrication of the SPM.

Task 2.2 6000/Series 60 Interface Unit Design

The 6000/Series 60 Interface Unit (IU) is required to provide the hardware interface for the SFEP into a large-scale host (6000/Series 60) computer system. The IU provides a local bi-directional communications channel up to 2000 feet in length between the SFEP bus and the Direct Channel connection of the 6000/Series 60 IOM.

The 6000/Series 60 IU design task includes the following subtasks: logic board design, backplane design, chassis design documentation, parts selection, development and product specifications, and support to IU evaluation tests. The output from the 6000/Series 60 IU design task will be the documentation which will permit fabrication of the 6000/Series 60 IU.

Task 2.3 SCOMP Design

The SCOMP design task encompasses the design activities required to modify a commercial minicomputer to interface with a SPM and IU and the preparation of test plans to be used in the SPM and IU evaluation.

The SCOMP design tasks include the following subtasks: the design and documentation of the CPU hardware and firmware modifications, the formulation of a master test plan, an SPM development test plan and procedures and an internal acceptance test plan.

The output from the SCOMP design task will be the documentation required to modify the CPU, test the SCOMP, and evaluate the SPM design.

Task 2.4 Test and Evaluation Software Design

Test and evaluation software is required to permit separate testing, verification, and evaluation of all functions designed into the SFEP hardware via the SPM and 6000/Series 60 IU.

This software will provide routines which exercise descriptors (cache), Descriptor Base Registers (DBR), all SPM control operations, and ring controls. In addition, test and evaluation software which exercises the security functions of the secure minicomputer will be developed and will support intraprocedure, interprocedure, intraprocess and interprocess context switching, I/O control and access control procedures. Additional software will exercise the operation of the 6000/Series 60 IU in a closed-loop (i.e., wrap-around) manner.

The test and evaluation software design task will be based on a software specification of the secure computer functions to be derived from the SFEP hardware Functional Specification. This software will be developed using the simulator defined in Task 3.5 and will form the basis for any diagnostic or acceptance test software developed. This task includes the design, coding, checkout, and documentation of the Test and Evaluation software.

Task 2.5 Test Equipment Design

The development of test equipment is required to permit evaluation and acceptance testing of the hardware subsystems.

Three items of test equipment are required:

a. History Memory

b. IU Wrap-Around Unit

c. TEMPEST Test Fixture

The History Memory is a module that plugs into the bus and selectively stores bus cycles. Used in conjunction with the Test and Evaluation (T and E) software, or any special diagnostic or evaluation software,, the History Memory provides a powerful diagnostic aide, particularly during initial debug or with low frequency failures, by providing a history of the data, address and controls of the bus cycles prior to and during an event.

The IU Wrap-Around Unit is used at the SFEP software development facility to provide the capability for dynamically testing the 6000/Series 60 IU. This unit will temporarily store data that is output to it, then transmit the data back to the SFEP for checking. Thus providing, in conjunction with the T and E software, a means of checking out the IU prior to integration with a host computer.

The TEMPEST test fixtures are plates and fixtures used during the TEMPEST qualification test (Task 2.12).

The test equipment task includes the design, documentation, fabrication and checkout of the test equipment specified.

### Task 2.6 Design Verification

This task is necessary to verify, with confidence, that the design objectives for security compromise probability which will be less than .000001 have been met. This task will also verify, with confidence, that the SCOMP hardware design functions exactly as specified (i.e., there are no security breach paths). This task includes two subtasks:

a. Probabilistic measure of security compromise due to hardware failures. This task resembles a "failure mode and effects" analysis on those components whose failure may jeopardize security. A baseline SFEP configuration is analyzed against defined hardware security breach categories.

b. Electrical and Logical design verification. This task will verify, with confidence, via computer simulations, computer analysis and/or algorithmic evaluations, that the electrical and logic design is complete.

The output from these tasks will be probabilistic and confidence measures. These outputs may dictate design modification and/or redundancy in critical paths. The need for and frequency of "health checking" software will be defined by the analysis. The analysis will be documented in a final report.

### Task 2.7 Communications Interface

This task will determine additional requirements for the communications interface. Additional studies will be performed which delineate hardware, logical, and software interfaces.

This task will study, as applicable, standard communications interfaces including AUTODIN II, the DCW communications controller and the ARPA networks resulting in a final report which summarizes any additional communications interface requirements.

### Task 2.8 SFEP Development Unit

This task will develop the first testable SFEP hardware.

The task consists of procurement, fabrication, assembly, and test of the SFEP development unit. The efforts included in this task are listed below:

a. SCOMP

   1. Procure minicomputer and peripherals

   2. Procure additional processor

   3. Perform modifications to processors

   4. Fabrication/assembly of two sets of SPM boards

   5. Assemble SCOMP

   6. SCOMP integration and test

b. 6000/Series 60 IU

   1. Procure chassis and power supply

   2. Fabrication/assembly of 6000/Series 60 IU boards

   3. 6000/Series 60 IU integration and test

c. System Integration

   1. Assemble SCOMP to 6000/Series 60 IU cables

   2. Integrate SCOMP and 6000/Series 60 IU

   3. Test

Completion of the design and development tasks (Tasks 2.1 - 2.5 above) will provide the necessary documentation to fabricate the SFEP development unit.

The output of this task will be a SFEP unit for use in the SFEP hardware and software development site and a vehicle for the SFEP demonstration.

Task 2.9 SPM Development Tests

This task consists of a series of tests which are conducted per the plans and procedures developed under Task 2.3. The purpose of these tests are to evaluate the utility, performance, and margins of the SPM design. Alternative implementations of various SPM functions will be assessed. The output from this task will be a test report and a preferred SPM design for printed circuit board development and Programmable Read-Only Memory (PROM) firmware.

Task 2.10 Printed Circuit Board Design

The objective of this task is to convert the SPM boards, that are wirewrapped or hardwired, into printed circuit boards prior to fabrication of the prototype unit.

This task will generate production documentation for printed circuit boards for the SPM. The basic subtasks are: layout, checking, digitizing, photo plotting, printed circuit board detail drawing, and producing assembly drawings. This task begins when the design and testing of the SPM boards are considered complete. The output of this activity is production documentation for the SPM boards.

Task 2.11 TEMPEST Compatibility Design

TEMPEST compatibility design is required to meet the Red/Black separation and other TEMPEST requirements.

This task includes the following subtasks: development of design requirements, strippers and filters design, printed circuit board design, chassis design, test software routines, and documentation. The output from this task will be the design documentation which will permit the fabrication of TEMPEST-related components.

Task 2.12 TEMPEST Qualification Test

This task will demonstrate compliance of the SFEP hardware to the TEMPEST requirements as defined in Task 2.6 above. The tests will be performed in simulated operational environments. Conducted and radiated emanations will be measured and correlated utilizing TEMPEST test equipment. This task will be based on a TEMPEST Test Plan.

The following subtasks are included: test plan development, qualifications test performance, and results analysis. Results of the qualification test and analysis will be documented in a test report.

Task 2.13 SFEP Prototype Unit

This task consists of the procurement, fabrication, assembly and test of the SFEP prototype unit. The SFEP prototype will be a ruggedized unit that includes TEMPEST hardware. This computer will be used for TEMPEST qualification testing and will be the SFEP unit delivered for eventual Multics integration.

The efforts included in this task are listed below:

a. SCOMP

    1. Procure minicomputer

    2. Perform modifications for ruggedization

    3. Fabrication/assembly of SPM boards

    4. Assemble SCOMP

    5. SCOMP integration and test

b. 6000/Series 60 IU

    1. Procure ruggedized chassis and power supply

    2. Fabrication/assembly of 6000/Series 60 IU boards

    3. Assemble 6000/Series 60 IU unit

    4. 6000/Series IU integration and test

c. Additional Hardware to Meet TEMPEST Requirements

    1. Fabricate special chassis

    2. Procure power supplies

    3. Procure components

    4. Procure circuit boards

    5. Assemble TEMPEST unit

    6. TEMPEST integration and test

d. System Integration

    1. Assemble (cable) SFEP prototype unit

    2. Test unit

Completion of the fabrication, assembly, and test portions of this task will result in a completed prototype hardware unit that forms a Secure Front-End Processor with TEMPEST capability.

Task 2.14 Delivered Unit Support

This task includes installation and integration of delivered equipment for use in the secure computer development site and includes unit installation and test, continued field support and maintenance, and reissued spare hardware materials.

Resource Requirements

The following table indicated the resource requirements necessary for the subtasks of Task 2.0.

MANPOWER
MANMONTHS/QUARTERS

PROJECT GUARDIAN
TASK 2.0 SECURE FRONT END PROCESSOR - HARDWARE

| TASK/DESCRIPTION | Quarters → | 1976 | | 1977 | | | | 1978 | | | | 1979 | | | | 1980 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 2.1 Security Protection Module Design | | 12.2 | 4.2 | 4.5 | 3.0 | | | | | | | | | | | | | | |
| 2.2 6000/Series 60 Interface Unit Design | | 3.4 | 5.7 | 8.8 | 4.5 | 1.5 | | | | | | | | | | | | | |
| 2.3 SCOMP Design | | 11.6 | 3.7 | 2.1 | | | | | | | | | | | | | | | |
| 2.4 Test and Evaluation Software Design | | 5.5 | 2.0 | 3.0 | | | | | | | | | | | | | | | |
| 2.5 Test Equipment Design | | 4.4 | 2.6 | 3.2 | 3.3 | .7 | .5 | .8 | | | | | | | | | | | |
| 2.6 Design Verification | | 5.3 | 4.4 | 5.9 | 1.0 | | | | | | | | | | | | | | |
| 2.7 Communications Interface | | | | | | 4.3 | 1.5 | | | | | | | | | | | | |
| 2.8 SFEP Development Unit | | 8.8 | 11.0 | 2.7 | 13.5 | | | | | | | | | | | | | | |
| 2.9 SPM Development Tests | | | | 4.0 | 1.0 | | | | | | | | | | | | | | |
| 2.10 Printed Circuit Board Design | | | | | | 7.0 | .5 | | | | | | | | | | | | |
| 2.11 TEMPEST Compatibility Design | | | | | .8 | 7.1 | 5.6 | | | | | | | | | | | | |
| 2.12 TEMPEST Qualification Test | | | | | | | | | 3.5 | .6 | | | | | | | | | |
| 2.13 SFEP Prototype Unit | | | .3 | .2 | .4 | 4.8 | 3.1 | 2.1 | | | | | | | | | | | |
| 2.14 Delivered Unit Support | | | | | | | | | 3.0 | | | 3.0 | | | | | | | |
| 2.15 Program Management (Task 2.0 and 3.0) | | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | | |

## Section III

## Secure Front-End Processor (SFEP) Software Development

This section describes the tasks required to develop the certifiably secure software system for a front-end communications processor. This software will consist of two major components: a general purpose security kernel suitable for secure communications processor applications, and the necessary operating system/application software to perform the Multics front-end processor functions.

To support the development of this software system and to accomplish the certification of the security kernel, it is necessary to define specification and programming standards, establish procedures and maintenance tools for certification, define modifications to existing programming languages, and develop test and evaluation methods for secure computer systems. Fortunately, many of these items can benefit from similar activities carried out for the Multics security kernel development. The SFEP software development will, where practical, use the same languages, standards, procedures, etc. as are used for Multics. Only where necessary will special standards and procedures be developed for SFEP.

The SFEP software development proceeds through a series of steps that allow measurement of progress against the proposed plan. First, formal specifications for the security kernel will be developed, reviewed with the Air Force and submitted to the certification process. In parallel with the certification process, a prototype kernel and SFEP operating system/application software will be implemented in assembly language. This assembly language software package will be used to demonstrate a free-standing capability. By this time the correspondence proofs of the formal kernel specifications will have been completed. Factoring in the lessons learned from the specification proof activity and from the prototype kernel development, a formal kernel will be developed in a modified PL/1 language. This kernel will be integrated with other SFEP software and used in the intermediate Multics demonstration that integrates Multics and the SFEP. Kernel certification (high level language) will be carried out in parallel with the integration under Task 1.27. Next, any changes required as a result of the certification and integration efforts will be made. This includes enhancements for performance and functionality. Finally, the SFEP is again integrated with Multics for the prototype demonstration.

The SFEP software development activity is subdivided into nine tasks. Task 3.1, 3.2 and 3.3 outline the required effort to produce the software for both the prototype and formal kernel. Tasks 3.4, 3.5 and 3.6 provide the necessary tools to support certification efforts and overall software program development. Tasks 3.7 and 3.8 describe the efforts required to produce

software that will suport a Multics front-end application. Task 3.9 is concerned with evaluating and testing the total software for both the prototype and formal systems. The effort described by these tasks is an estimate of what is required to develop a certified secure SFEP software system. It is expected that as the certification technology evolves, changes may be required to the described approach.

Task 3.1 Top Level Kernel Specifications

This task will produce the formal specifications for a general purpose security kernel suitable for secure communications processor applications. Both the top level specification and the lower level hierarchical specifications will be generated.

The impact on efficiency and certifiability of SFEP architecture, communications processing functions and the SFEP to Multics interface, will be factored into the development of these specifications. However, where practical, these specifications will be similar to the Multics kernel specifications, both in functions covered and in the specification language used. Of course, the SFEP kernel functionality is expected to be less than that of Multics.

The top level specification (TLS) covers the kernel's interface with the operating system/applications software. The preliminary version, submitted in February, 1976, for Air Force review, will form the basis for work in this area. Additional effort under this task includes the revision of the TLS on the basis of Air Force review, identification of trusted processes, and resolution of open issues.

It is assumed that trusted processes are outside the distributed kernel, but are included within the security perimeter. That is, they must be proven correct as part of the certification task. For convenience, the development of trusted processes are included in their respective functional areas, i.e., either as part of the operating system or the communication subsystem.

The TLS will be extended to lower levels to include all modules in the security perimeter (i.e., distributed kernel and trusted processes).

To achieve a reasonable development schedule, work on the lower level specifications must start before the top level correspondence proofs have been completed. The risk in this early start (due to certification related changes) will be minimized by submitting the top level specifications for Air Force review prior to starting on the lower level specifications. A formal Preliminary Design Review (PDR) will be held after completion of the top level correspondence proofs done in Task 3.4.

These formal specifications will serve as the basis for the development specification function for both the prototype and formal kernel development.

Task 3.2 Prototype Kernel Development

This task will implement and test the prototype security kernel for the SFEP in assembly language. This prototype will be integrated with Operating System software developed under Task 3.7 and used in the free-standing SFEP demonstration. The free-standing SFEP will be used for evaluation testing of the kernel. The prototype will serve as the model for the formal kernel to be implemented in Task 3.3.

The prototype kernel implementation will be based on the formal specifications developed under Task 3.1. A detailed design will be carried out and documented as a preliminary Product Specification. This will include a description of the kernel structure and functions, the data base, the individual program modules or components, and a detailed definition of the interface between the kernel and operating system/application software.

The prototype kernel will be implemented and tested. Test cases will be developed to check out the kernel and to allow its use in the free-standing SFEP demonstration. After the demonstration, the prototype kernel Product Specification will be finalized.

The standards established in Task 3.6 will be followed during the development of the prototype kernel. This will allow the prototype kernel design to be used for the formal kernel. Some differences between the formal and prototype kernel designs are, of course, anticipated as a result of the certification and test/evaluation activities.

Task 3.3 Formal Kernel Development

This task will implement the SFEP kernel in the system programming language defined under Task 3.6. This kernel will be integrated with other SFEP software and used in the intermediate Multics demonstration that integrates Multics and the SFEP.

The formal kernel will be based on the specifications developed under Task 3.1 and certified under Task 3.4. It is expected that both the certification effort and the prototype kernel testing will identify changes that must be made to the specifications. These changes will be made under this task, the certification will be updated under Task 3.4.

After this update is completed, the detailed design for the prototype kernel will be updated to reflect the revisions and to include the full functionality of the formal specifications. The

latter includes functions such as the Multics kernel interface. The detailed design will be incorporated into a preliminary Product Specification for the formal kernel. This specification will be submitted to the Air Force for review. After the Air Force review, the kernel will be implemented in the system programming language, compiled (using the compiler developed under Tasks 1.1 and 3.5) and tested. It is then ready for use in the intermediate Multics demonstration. Formal kernel development and modification will be performed by cleared personnel within a secure environment similar to that in which the Multics kernel development similar to that in which the Multics kernel development will take place.

Task 3.4 Certification Procedure Development

This task addresses the effort required for technical certification of the SFEP kernel software. The effort is subdivided into four main activities: 1) establishing procedures and tools for certification and maintenance of certification; 2) on-going review of the kernel development to assure that the specifications and design are proceeding in a direction that allows certification, 3) the actual technical certification, (i.e., correspondence/correctness proofs), and 4) the maintenance of these proofs. To maximize the probability of success, the SFEP certification effort will be coordinated with the Multics effort and, where feasible, commonality with Multics techniques and tools will be maintained.

The procedures and tools to be used for the technical certification and maintenance of this certification will be established. It is expected that automated tools will not be used for the certification activity; however, some might be appropriate for the maintenance effort. Procedures and tools established for Multics will be reviewed and used as appropriate. The chosen approach will be shown to be sufficient, will be documented, and a plan prepared for carrying out the certification.

The certification team will carry out an on-going review of the kernel development effort. This will increase the probability of success in the certification effort by minimizing the use of inappropriate techniques.

The actual certification will be carried out in three steps. First, the correspondence proof between the kernel top level specification and the Air Force security model will be developed; next, the correspondence between intermediate levels of abstraction will be proven; and finally, (under Task 1.27) the proof of correctness of the system programming language representation will be attempted. A similar process will be carried out for the trusted processes. However, it is expected that the certification of the trusted processes will require less

effort than distributed kernel certification. The proofs at each step will be documented and submitted to the Air Force for review.

The correspondence proofs will be updated if major changes are made to the specifications or programs.

A final technical report on the certification effort will be prepared at the end of the project.

Documents and software used in the certification process will be safeguarded consistant with configuration management procedures (Task 4.2).

Task 3.5 Software Development Tools

This task will ensure that the necessary language and software development tools are available for the development of SFEP software. Items included are: assembler, simulator, system programming language, compiler, loader and an interactive debug routine.

Requirements for the software development tools are based on following a two-step approach. Initial development of SFEP software will be done on a Multics host resident system consisting of an assembler, simulator, and (for the formal kernel) compiler. This will allow interactive software development, using the Multics based software for source edits, file maintenance, assemblies and debug and check-out via the simulator. Use of the host resident software will also minimize schedule conflicts for the use of SFEP development hardware. Without the host resident software, multiple software developers would be competing with each other and the hardware testing and evaluation activity for use of the SFEP hardware. Final software checkout must, of course, be done on the actual SFEP hardware. Thus, some native mode software will also be required. This latter will be based on existing minicomputer software modified to accommodate the SPM. The software development items are discussed in the following paragraphs.

Assembler - The existing Multics resident assembler will be modified to handle the added functions of the SPM. The existing assembly language manuals will be updated to reflect these changes. This assembler will be used in development of the prototype kernel, the operating system and the communications subsystem.

Simulator - The existing Multics resident interpretive simulator for the selected minicomputer will be modified to include the SPM functions. A user's manual will be prepared. The simulator is required to allow: SFEP hardware evaluation prior to availability of the actual hardware, development of SFEP hardware

test and verification software, analysis of probabilistic measure of security compromise (Task 2.6), and Multics host resident SFEP software development as described above.

System Programming Language and Compiler - To allow certifiability of the SFEP kernel, a higher level language must be used as an intermediate step in the kernel development. It is intended that a common language be used for the SFEP and Multics kernels. The degree of commonality will be determined under this task through review of the language defined for Multics (a modified PL/I) under Task 1.1. If a 100% commonality is not possible because of hardware architecture differences, the necessary changes to the Multics language will be defined under this task.

A compiler will be developed to translate this language into the SFEP assembly language. Under this task, a code generator for the SFEP will be developed and integrated with the front-end (syntax analyzer) of the compiler developed under Task 1.1.

Loader and Debug - The existing minicomputer loader and interactive debug programs will be modified to work with the SFEP. User documentation will be updated.

Task 3.6 Formal Programming Technology

This task will establish standards and ensure that they are followed throughout the course of the project.

To assure that SFEP kernel software certification can be accomplished, standards will be established and documented for formal specifications and programming conventions. Standards established for Multics will be reviewed and used as appropriate. Necessary training to familiarize personnel with these standards will be carried out.

Task 3.7 Develop OS Functions for Multics SFEP

This task will develop the supervisor and service functions that are required, in addition to the communications subsystem and kernel, to develop and demonstrate the Multics SFEP.

The supervisor functions consist of the resource allocation "policy" functions that do not impact security. While the specifics of functionality and implementation remain to be defined, two basic functions are planned. These are a process scheduler and a memory manager. The scheduling function will establish process priority and communicate its scheduling decisions to the kernel. The memory manager's primary functions will be the housekeeping of free memory and buffers. The actual dispatching and allocation of physical resources will be done by

the kernel. The classification of some of these supervisor functions as system high routines will be done as part of this task. Three other functions normally in a supervisor (clock, interrupt and trap management), are to be handled by the kernel.

Other functions to be implemented under this task include: an operator interface, input/output device drivers, and a bootload function.

These functions are included primarily to support integration and test activity. However, they are all consistent with the "general purpose" SFEP objectives. The peripheral devices include: a disk (to support software development and test), a diskette (for bootloading), and an operator/maintenance console (for troubleshooting). The bootload function is required for free-standing operation and for troubleshooting. Specific functionality and implementation details of these functions will be defined as part of the design task.

Two important omissions should be noted. First, the SFEP will not support peripherals for Multics. Second, automatic recovery from failures is not addressed. The recovery problem is assumed to be outside the scope of this effort.

The design of the OS functions will be based on constraints imposed by the kernel formal specifications and on the kernel to OS interface specified by the prototype kernel design in Task 3.2. Based on these constraints, a development specification will be prepared for the OS. Detailed design will be carried out and implemented in assembly language. A product specification for the OS will be prepared.

### Task 3.8 Develop SFEP Communications Subsystem

This task will develop the communications subsystem for the Multics SFEP. Two basic functions will be included. These are a terminal handler and a Multics/SFEP interface. Some elements of these functions will be trusted processes.

The communications subsystem design requirements will be based on many constraints. These include: kernel formal specifications, detailed kernel design, agreed-to Multics/SFEP interface protocol, detailed operating system design, and Multics terminal and network interfaces. The definition of functional and interface requirements will be coordinated with both the Multics design team and the Air Force.

The development of the communications subsystem includes: the preparation of development specifications, detailed design, preparation of product specifications, implementation, integration with other SFEP software, and testing. The resulting software system will be used in the intermediate Multics/SFEP demonstration.

Task 3.9 SFEP Integration and Test

This task will unite the SFEP hardware and software into a single system, integrate it with a Multics host, and demonstrate that it can perform the Multics front-end processing functions. This will be accomplished in three steps: progressing from a free-standing SFEP system, through an intermediate Multics/SFEP system, to the prototype Multics/SFEP system.

The free-standing SFEP system will demonstrate the operation of SCOMP hardware and kernel software. This system will be established by integrating the SCOMP hardware from Task 2.0 with the following SFEP software elements: prototype kernel from Task 3.2 and operating system software from Task 3.7. The free-standing system will also be used for performance evaluation.

The intermediate Multics/SFEP system will demonstrate the ability of the SFEP to interface with Multics and perform its front-end functions. For this demonstration, the formal kernel (still uncertified at this point in time) from Task 3.3 will be used. The communications subsystem software from Task 3.7 will be used to handle the Multics interface and the terminal subsystem.

The final demonstration will incorporate the enhancements and changes made to the software as a result of the previous demonstrations and the certifications activity.

Prior to each of the above three demonstrations, test plans and procedures will be prepared and submitted to the Air Force for review. A test report will be prepared after each of the demonstrations. The final report for all of the SFEP software activity will also be prepared under this task.

Resource Requirements

The following table indicates the resource requirements necessary for the subtasks of Task 3.0.

MANPOWER
MANMONTHS/QUARTERS

PROJECT GUARDIAN
TASK 3.0 SECURE FRONT END PROCESSOR - SOFTWARE

| TASK/DESCRIPTION | QUARTERS → | 1976 | | 1977 | | | | 1978 | | | | 1979 | | | | 1980 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 3.1 Top-Level Kernel Specification | | 6.0 | 6.0 | 6.0 | | | | | | | | | | | | | | | |
| 3.2 Prototype Kernel Development | | 3.0 | 6.0 | 6.0 | 9.0 | 9.0 | 6.0 | | | | | | | | | | | | |
| 3.3 Formal Kernel Development | | | | | | | | | | 3.0 | 6.0 | 6.0 | 3.0 | 3.0 | 6.0 | 6.0 | 3.0 | | |
| 3.4 Certification Procedure Development | | 6.0 | 7.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | | |
| 3.5 Software Development Tools | | 10.5 | 8.5 | 7.5 | 1.5 | | | | | | | | | | | | | | |
| 3.6 Formal Programming Technology | | 4.5 | 5.5 | 4.5 | 4.5 | 3.0 | 3.0 | | | | | | | | | | | | |
| 3.7 Develop OS Functions for Multics SFEP | | 10.5 | 10.5 | 10.5 | 10.5 | 7.5 | 4.5 | 4.5 | 4.5 | 4.5 | 4.5 | | | | | | | | |
| 3.8 Develop SFEP Communications Subsystem | | | | 6.0 | 6.0 | 6.0 | 8.0 | 6.0 | 6.0 | 6.0 | 8.0 | | | | | | | | |
| 3.9 SFEP Integration and Test | | | | | | | 2.0 | 6.0 | 8.5 | 10.5 | 6.0 | 7.5 | 8.5 | 9.5 | 6.0 | 6.0 | 5.0 | | |

# Section IV

# Project Control

## Introduction

This task is the central focal point for control and support of all technical and management activities in the project. Through management support, this task initiates all technical and management review activities for all aspects of the project. All project documentation is also centralized and coordinated by this task. To support all system development requirements, the Project Control task also has responsibility for providing computer system resources on a regular basis.

This task will provide continuous and efficient support to all project activities. Any identified problems anywhere in the project will be resolved by personnel assigned to this task. To achieve a smooth, efficient project, this task will provide a coordinated flow of all project documentation to the Air Force.

A Multics service system will provide the administrative tools of document preparation, editing, program schedule maintenance, and other administrative efforts. When mutually agreeable, all published, unclassified documents produced by this project can be printed at any time as they will be available for common perusal by appropriate Air Force and Honeywell personnel. In this way, automated project control procedures will be an integral part of this program.

## Task 4.1 Project Management

This task will integrate all efforts of the contractor and subcontractor organizations. The Project Management task will coordinate the intermediate technical milestones which are required to support project decision points for the Air Force. Specific subtasks are:

a. Preparation and update of a continuous program schedule for Air Force review

b. Preparation and scheduling of formal Technical Review meetings.

In addition to the technical review meetings, Project Management will conduct regular montnly management review meetings and, as they become necessary, informal technical working meetings.

Project Management will ensure that program objectives are accomplished in a timely and cost-effective manner. This task impacts all aspects of the project and includes the following

subtasks:

a. cost control

b. schedule control

c. status report control

Task 4.2 Configuration Management

This task establishes procedures and controls for configuration management and coordinates all changes to authenticated configurations. Configuration control will be the formal means of ensuring the integrity of the system and success of the project. A plan will be produced that defines policies and procedures for configuration control.

The configuration management task will provide configuration control over the following items:

a. Prototype Secure Multics

b. Security Protection Module (allocated)

c. Honeywell 6000/Series 60 Interface Unit

d. Multics IOM Modifications

e. Multics Security Kernel

f. SFEP Security Kernel

Task 4.3 Data Management/Documentation

This task provides all the data management (i.e., documentation) to coordinate and monitor the progress of preparation, review and distribution of project documentation. Outputs from this task are as follows:

a. Technical Status Reports

b. Technical Specifications

c. Technical Notes

Task 4.4 Computer Resources - Multics Service Site

The Multics service site provides the initial working host system on which program development and unit (isolated) check-out of software will occur. The Multics service system is a nonsecure site and provides library copies of all current system software

and project documentation. Initial design and coding of system software is performed on the service system. After the software is developed and meets the designer's satisfaction as to unit checkout, it is then ready for trial integration on the Multics development computer.

The nonsecure Multics service site will be MIT's Information Processing Center in Cambridge, Massachusetts with supplemental service provided by Honeywell's Multics facility in Phoenix, Arizona.

Use of the Multics service system is in a shared nondedicated environment. There may be many other simultaneous users of the service system, some of which may not be utilizing the system for efforts related to this project.

### Task 4.5 Computer Resources - Multics Development Sites

The Multics development sites provide computing facilities in which software development and system test can be undertaken. There are two different development sites presently identified and required for this project. These are the nonsecure Multics development system at Honeywell's Cambridge Information Systems Laboratory and the secure development site at Hanscom Air Force Base, Bedford, Massachusetts. The secure development site will be established specifically for the purpose of supporting system development efforts in a cleared facility.

### Task 4.5.1 Computer Resources - Nonsecure Development Site

The nonsecure Multics development system at Honeywell's Cambridge Information System Laboratory, Cambridge, Massachusetts, will be used for software check-out in an integrated system test mode in an uncleared facility. Unlike the service system, the nonsecure development system is used in a dedicated, block-time environment. Normally, users other than the designer responsible for the specific integration/system test run are not permitted simultaneous access to the development system. Each computer run is identified by a sign-up sheet for a block of time.

Due to the very nature of block-time usage, the Multics service site cannot be the same as the Multics development site. The service system provides regular continuous service and extreme scheduling difficulties arise in trying to intermix development block-time requirements with continuous undisrupted service requirements.

### Task 4.5.2 Secure Development Site

Prior to the recoding of the Multics and SFEP security kernels, the secure Multics development system must be established. This secure site is required so that the kernel development for the operational prototype secure Multics demonstration and the final SFEP can be performed in a Top Secret controlled location to maintain integrity of the security kernels. The secure Multics development system will be installed in government-furnished facilities. The establishment of this site requires specially-trained site support and operations personnel. Task 1.30 has defined the preparation of the secure site. Task 1.29 has described the task of developing a team of systems designers who will perform the software development of the security kernels for the operational prototype secure Multics demonstration and the final SFEP in the secure development site. These system designers are separate and distinct from the site and operations support personnel required to maintain continuous operation of the secure development site.

The secure development site is an interim facility for integrated system test, and preliminary test and evaluation, of hardware and software components, prior to the eventual installation of the secure Multics at an operational site such as the Air Force Data Services Center (AFDSC). The installation of the secure Multics at this operational site is a major goal of this Integration Plan but has not been included in the resource estimates provided, because the actual installation will occur after the five-year scope of this plan.

### Task 4.6 SFEP Resources

Paralleling the Multics development sites is the nonsecure SFEP development site at Honeywell's Aerospace facility. This site will provide all computer resources necessary for the development of the SFEP software. The nonsecure SFEP system will be used for both the unit check-out and integrated check-out of all SFEP software. A second SFEP system will be installed in the secure Multics development site described in Task 4.5.2 above to support the operational prototype Multics demonstration.

### Resource Requirements

The following table indicates the resource requirements necessary for the subtasks of Task 4.0.

MANPOWER
MANMONTHS/QUARTERS

PROJECT GUARDIAN
TASK 4.0 PROJECT CONTROL

| TASK/DESCRIPTION | Man Months Quarters → | 1976 | | 1977 | | | | 1978 | | | | 1979 | | | | 1980 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 4.1 Project Management | | 15.0 | 15.0 | 15.0 | 15.0 | 15.0 | 15.0 | 15.0 | 15.0 | 15.0 | 15.0 | 15.0 | 15.0 | 12.0 | 12.0 | 12.0 | 12.0 | 12.0 | 12.0 |
| 4.2 Configuration Management | | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 4.3 Data Management/Dcoumentation | | 6.0 | 6.0 | 9.0 | 9.0 | 9.0 | 9.0 | 9.0 | 9.0 | 9.0 | 9.0 | 9.0 | 9.0 | 6.0 | 6.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| 4.4 Computer Resources* - Multics Service Site (GFE) | | 68.5 | 63.3 | 82.8 | 82.8 | 69.4 | 80.5 | 90.4 | 85.1 | 54.1 | 48.0 | 20.3 | 4.0 | 4.2 | 4.2 | 3.8 | 4.0 | 4.3 | 4.3 |
| 4.5 Computer Resources* - Multics Development Sites | | | | | | | | | | | | | | | | | | | |
| 4.5.1 Non-Secure Development Site | | 86.4 | 66.4 | 113.6 | 120.3 | 135.6 | 290.5 | 271.8 | 235.8 | 260.0 | 228.0 | 40.5 | 40.0 | 40.0 | | | | | |
| 4.5.2 Secure Development Site | | | | | | | | | | 1.8 | 1.5 | 205.3 | 316.8 | 334.3 | 318.4 | 318.4 | 283.9 | 184.3 | 253.5 |
| 4.6 SFEP Resources | | | | | | | | | | | | | | | | | | | |

## Section V

## Summary and Conclusions

This report describes the problem of multilevel computer security, reviews past efforts in the field. The earlier programs have carried the reference monitor concept from an academic abstraction to a limited security kernel. An Integrated plan has been presented in this report to carry the security kernel concept into a large-scale computer system. The plan is composed of three major development tasks and one support task. The eventual goal of these tasks is to demonstrate the feasibility of attaining a certifiably secure Multics service. This program will attain the goal of an open, multilevel, secure computer system.

Task 1.0 is the mainline task of the project. It integrates the results of the other tasks and provides the catalyst to attain the program goal. The end result of Task 1.0 will be a demonstration that the program goals have been met. Other benefits are a greatly increased understanding of the problem of computer security and a well-developed methodology for specifying future secure computer systems.

Tasks 2.0 and 3.0 support Task 1.0, supplying the required communications capability. The end result of these tasks will be a certifiable secure communications front-end processor for use with Multics. Other benefits are increased understanding of the problem of communications processor security and a well developed methodology for specifying future secure communications processors.

Task 4.0 supports the other three tasks. The primary function of Task 4.0 is to ensure the coordinated management of the program. The principle long term benefit is an integrated collection of program documentation.

The resulting prototype Multics computer system will demonstrate the ability of the military to process information of mixed security levels. This will be a major step toward the military's capability to make secure multilevel on-line systems operational and should result in the saving of a large portion of the annual cost of computation. Other agencies will benefit also, as the availability of secure computer systems will increase their ability to build data processing systems that truly safeguard information and help to ensure the individual's rights to privacy.

Summary of Resource Requirements

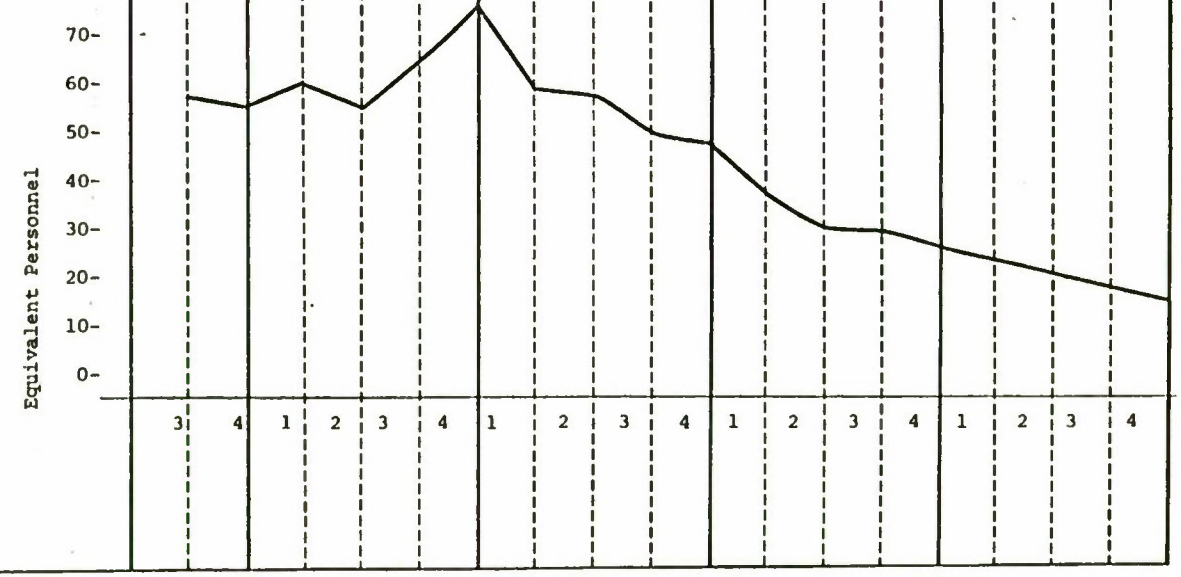
The following table summarizes the resource requirements necessary to perform all the tasks described in this report.

PROJECT GUARDIAN
TOTAL RESOURCE REQUIREMENTS

Months/Quarter

| TASK/DESCRIPTION | Manmonths Quarters→ | Period 1 | | Period 2 | | | | Period 3 | | | | Period 4 | | | | Period 5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| o Development of a Prototype Multics Kernel | 1226.5 | 52.5 | 61.5 | 72.5 | 74.0 | 102.0 | 121.0 | 117.0 | 110.0 | 96.0 | 90.0 | 60.0 | 45.0 | 48.0 | 42.0 | 59.0 | 39.0 | 39.0 | |
| o Secure Front End Processor - Hardware | 285.6 | 54.2 | 86.9 | 37.4 | 30.5 | 28.4 | 54.2 | 15.9 | 9.5 | 3.6 | 3.0 | 4.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | | |
| o Secure Front End Processor - Software | 400.5 | 40.5 | 43.5 | 43.5 | 34.5 | 28.5 | 26.5 | 19.5 | 22.0 | 27.0 | 27.5 | 16.5 | 14.5 | 15.5 | 15.0 | 15.0 | 11.0 | | |
| o Project Control | 432.0 | 24.0 | 24.0 | 27.0 | 27.0 | 27.0 | 27.0 | 27.0 | 27.0 | 27.0 | 27.0 | 27.0 | 27.0 | 21.0 | 21.0 | 18.0 | 18.0 | 18.0 | |
| Total Project Man Months | 2344.6 | 171.2 | 165.9 | 180.4 | 166.0 | 185.9 | 228.7 | 179.4 | 168.5 | 153.6 | 147.5 | 108.0 | 88.0 | 86.0 | 79.5 | 73.5 | 69.5 | 57.0 | |
| Equivalent Personnel | | 57.1 | 55.3 | 60.1 | 55.3 | 62.0 | 76.2 | 59.8 | 56.2 | 51.2 | 49.2 | 36.0 | 29.3 | 28.7 | 26.5 | 24.5 | 23.2 | 19.0 | |

o Resources Loading



# Appendix A

## Supporting Efforts

### Introduction

Earlier sections of this report have identified and described the four major tasks involved in the design and development of a secure Multics. This Appendix identifies and describes a set of supporting efforts that may be followed in order to augment the secure Multics and to extend its utility. These supporting efforts are in addition to those required for the major tasks described in the main sections of this Integration Plan. They cover desirable capabilities, but are not essential to meet the principle goals of Project Guardian.

Four supporting efforts have been identified at this time. It is expected that additional efforts will be defined as the project proceeds. The supporting efforts described below are:

A. Secure Data Base Management System

B. Cryptographic Multiplexor

C. Secure Office Terminal

D. Security Audit and Surveillance

Each is described in a separate section below.

Supporting Effort A: Secure Data Base Management System

## Section A1.0 INTRODUCTION AND OVERVIEW

This supporting effort addresses the design, development, implementation, and possible certification of a Data Base Management System (DBMS) which, when operating on the secure Multics, would provide the capability for a data base to contain data of several classification levels and categories and be accessible by users of various clearance levels and categories while guaranteeing that no unauthorized data accesses can occur.

There are several possible approaches to the solution of this problem, namely:

(1) The design proposed by Hinke and Schaefer, of System Development Corporation (SDC); (1)

(2) An extension of the Multics Data Base Manager (MDBM) currently under development by Honeywell, the initial version of which is to be included in MR 4.0. Such an extension would be necessary in order to provide multilevel access control at the attribute (field) level. Currently, the MDBM provides access control only at the relation (record type) level. There are two such extensions currently under consideration. One approach would require that the MDBM contain security sensitive code, but may provide superior performance. Another approach may not require security sensitive code, but may also result in degraded performance; although the degradation should not be as severe as would be expected for the SDC proposal.

The basic difference among all approaches concerns the data base storage format and the resulting effects upon performance, compatability with the standard product, and the possibility of the existence of security sensitive code within the DBMS.

It is therefore proposed that this task commence with a study to evaluate the relative merits of these approaches and to choose from among them. If the SDC proposal is chosen, the study would be followed by tasks to design and implement the proposed primitive functions and to incorporate them into the MDBM. If an Honeywell approach is chosen, the following tasks would be to design the specified security modifications and incorporate them into the MDBM. If it is necessary for security sensitive code to exist within the MDBM, then the programming language and

(1) Thomas H. Hinke and Marvin Schaefer, "Secure Data Management System", RADC-TR-75-266, System Development Corporation, Santa Monica CA, November 1975.

methodology developed in major Tasks 1.1, 1.5, and 3.4 would be utilized to re-implement and prove the correctness of the security sensitive code. The task schedules show an estimate of the manpower required to perform the effort for both approaches.

## Section A2.0 TASK PLAN

### Task A1.0 Study to Evaluate Alternatives

The goal of this task is to choose between the data base structure proposed by SDC and that defined for the MDBM. Both are relational structures, but differ in that the SDC proposal requires that relation attributes be stored in separate files, while the MDBM places an entire relation in a file. One result of this difference is that the SDC approach may cause a significant performance degradation because of the larger number of segments required to be active while accessing the data base. However, there is a possibility that the inclusion of security sensitive code may be required within the MDBM, in order to provide multilevel security at the attribute (field) level.

The following tasks will be performed during this study:

(1) A comparison between the performance of the MDBM and that of an attribute-per-segment relational implementation will be performed. One approach may be to use an existing attribute-per-segment implementation such as the Relational Data Management System (RDMS) at MIT. Another approach may be to perform a rudimentary implementation of the SDC proposed primitives.

(2) A determination of the amount of security sensitive code within the MDBM required to provide multilevel security at the attribute (field) level will be made for all Honeywell approaches. There is a high probability that, in one approach, none would be required.

(3) An evaluation of the impacts of all approaches with regard to standard product compatability will be performed.

(4) Based on the results of the above tasks, a decision will be made as to which approach the actual implementation is to follow. A final technical report containing the rationale for the decision, together with specifications for the design and implementation of the chosen approach, will be produced. This report will be the subject of a design review at the completion of this task.

At this point in the task, one of two paths will be followed, depending upon the decision resulting from Task A1.0.

Task A2.0 SDC Approach

The following three subtasks are required if the SDC approach is chosen.

Task A2.1: Design of the SDC Proposed Primitives

This task will involve the design of those primitives defined in the SDC report, and identification of the points of interface with the MDBM. The output from this task will be a design document. Further work will be contingent upon the approval of the design at a design review to be held at the time this task is completed.

Work will proceed concurrently along two fronts:

(1) the design of the data manipulation and data description primitives specified in the SDC proposal.

(2) the identification and specification of the interfaces between the primitives and the MDBM. This will involve all points where the MDBM is to communicate with the database, the data descriptions, and the storage structure.

Task A2.2: Implementation of the Primitives

This task will begin after successful completion of the above design review. It will involve the development of a Multics space manager conforming to the suggestions in the SDC report. The result will be a set of primitives coded in PL/I which will be utilized by all data description and data manipulation functions of the MDBM.

Work will proceed concurrently along two paths:

(1) the development of the data manipulation primitives as described in section 4.3.1 of the SDC report.

(2) the development of the data description primitives mentioned in section 4.3.2 of the SDC report.

Task A2.3: Incorporation of the Primitives Into the MDBM

This task will involve the incorporation into the MDBM of all the primitives developed in the previous task. This task includes testing and performance tuning. Those

portions of the MDBM identified in Task A2.1 will be rewritten to accomodate the primitives and the proposed storage structure. The output of this task will be a demonstrable version of the MDBM which uses the primitives for all data description and data manipulation functions.

At this point, Task A is completed if the SDC approach was followed.

Task A3.0 Honeywell Approach

The following describes the subtasks required if one of the Honeywell approaches is followed.

Task A3.1: Design of the MDBM Security Modifications

The modifications to the MDBM necessary to implement a multilevel security capability will be designed in accordance with the specifications produced by Task A1.0. The output of this task will be a technical design memo.

Task A3.2: Implementation of the MDBM Security Modifications

The MDBM security modifications, as designed in the previous task, will be implemented. This task includes testing and performance tuning. They will be coded in PL/I. The output of this task will be a demonstrable version of the MDBM which incorporates the security modifications. If the security modifications do not require the use of security sensitive code within the MDBM, this will be the final product.

The following subtasks are required only if an Honeywell approach requiring security sensitive code within the MDBM has been chosen.

Task A3.3: Reimplementation of MDBM Security Sensitive Code

If the security modifications require the use of security sensitive code within the MDBM, this code will be reimplemented, using the System Programming Language developed as part of major Task 1.1, for inclusion as part of the third Multics prototype.

Task A3.4: Proof of MDBM Security Sensitive Code

If the security modifications require the use of security sensitive code within the MDBM, the techniques developed under major Tasks 1.5 and 3.4 will be used to begin the proof of this code. The magnitude of this task will be dependent upon the then-current state of the art of program correctness proofs. A review of the progress of this task will be held about three months after its start.

PROJECT GUARDIAN
SECURE DATA BASE MANAGEMENT SYSTEM

Task A

| Task/Description | Quarters → | Period 1 | | | | Period 2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| NOTE: SDC Proposed Implementation | | | | | | | | | |
| A1.0 Study to Evaluate Alternatives | | | | | | | | | |
| A2.1 Design of SDC Proposed Primitives | | | | | | | | | |
| A2.2 Implementation of the Primitives | | | | | | | | | |
| A2.3 Incorporation of the Primitives into the MDBM | | | | | | | | | |

MANPOWER
MANMONTHS/QUARTER

PROJECT GUARDIAN
SECURE DATA BASE MANAGEMENT SYSTEM

Task A

| Task/Description | Quarters → | Period 1 | | | | Period 2 | | | | Period 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| NOTE: SDC Proposal Implementation | | | | | | | | | | | | | |
| A1.0 Study to Evaluate Alternatives | 18.0 | 9.0 | 9.0 | | | | | | | | | | |
| A2.1 Design of SDC Proposed Primitives | 20.0 | | | 12.0 | 8.0 | | | | | | | | |
| A2.2 Implementation of the Primitives | 8.0 | | | | 4.0 | 4.0 | | | | | | | |
| A2.3 Incorporation of the Primitives into the MDBM | 16.0 | | | | | 8.0 | 8.0 | | | | | | |

PROJECT GUARDIAN
SECURE DATA BASE MANAGEMENT SYSTEM

Task A

| Task/Description | Quarters → | Period 1 | | | | Period 2 | | | | Period 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| NOTE: Extension to the Honeywell Multics Data Base Manager Approach | | | | | | | | | | | | | |
| A1.0 Study to Evaluate Alternatives | | | | | | | | | | | | | |
| A3.1 Design of MDBM Security Modifications | | | | | | | | | | | | | |
| A3.2 Implementation of MDBM Security Modifications | | | | | | | | | | | | | |
| A3.3 Reimplementation of the MDBM Security Sensitive Code (if necessary) | | | | | | | | | | | | | |
| A3.4 Proof of MDBM Security Sensitive Code (if necessary) | | | | | | | | | | | | | |

MANPOWER
MANMONTHS/QUARTERS
Task A

PROJECT GUARDIAN
SECURE DATA BASE MANAGEMENT SYSTEM

| Task/Descriptions | Manmonths | Period 1 | | | | Period 2 | | | | Period 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Quarters → | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| NOTE: Extension to the Honeywell Multics Data Base Manager Approach | | | | | | | | | | | | | |
| A1.0 Study to Evaluate Alternatives | 18.0 | 9.0 | 9.0 | | | | | | | | | | |
| A3.1 Design of MDBM Security Modifications | 20.0 | | | 12.1 | 8.0 | | | | | | | | |
| A3.2 Implementation of MDBM Security Modifications | 24.0 | | | | 4.0 | 12.0 | 8.0 | | | | | | |
| A3.3 Reimplementation of the MDBM Security Sensitive Code (if necessary) | 24.0 | | | | | | | 12.0 | 12.0 | | | | |
| A3.4 Proof of MDBM Security Sensitive Code (if necessary) | 48.0 | | | | | | | | | 12.0 | 12.0 | 12.0 | 12.0 |

## Supporting Effort B: Cryptographic Multiplexor

### Section B1.0 INTRODUCTION AND OVERVIEW

This supporting effort considers and plans for the interface between the Secure Communications Controller (SCC) and the Secure Front-End Processor (SFEP). The Secure Communications Controller is being developed under the sponsorship of the Communications Security Engineering Office (ESD/DCW). (1) The SCC integrated controller and communications security (cryptographic) functions. The SCC will be capable of simultaneously servicing several remote terminals and will eliminate the need for one communications security (cryptographic) device at the computer site for each remote site. The SCC may be considered as a cryptographic multiplexor.

The planned activities for accomplishing the interface are delineated in the task plan of Section B2.0. The overall goal of the plan is to develop a prototype unit, based on in-depth studies, which will establish a design capable of an easy transition into a production phase. Although the main thrust of the plan is an optimized prototype unit, a low-cost parallel effort (described in the plan) may be required to support the study activities. The task schedule shows an estimate of manpower requirements to accomplish the study and development of the proposed SFEP/SCC interface.

### Section B2.0 TASK PLAN

#### Task B1 SFEP/SCC Interface Integration Studies

This task shall consist of defining functionalities and requirements of both the SFEP and SCC from the standpoint of supporting multiple secured terminal devices through communication links. The output of this task shall result in sufficient understanding and definition of requirements to implement an optimum configuration study task (Task B5) and to decide if the development of a functional evaluation unit (Task B3) would be advantageous as a vehicle to support the optimum study task. The implementation of the evaluation unit tasks shall be contingent upon the results of these initial studies and upon Air Force approval.

---

(1) "Secure Communications Controller", Project 7820/01/01.

Task B2 Definition of a SFEP/SCC Functional Evaluation Unit

This task shall define the degree of functionality of the Functional Evaluation Unit. Available hardware elements and hardware design tasks shall be determined according to the functionality requirements. Strong consideration shall be given to incorporation of off-the-shelf or GFE hardware elements to achieve a low-cost implementation. In addition to hardware selection, test and evaluation (T&E) software approaches and techniques shall be established.

The present visibility of a functional unit indicates its implementation may be accomplished by utilization of a prototype SFEP with a Multi Line Communication Controller (MLCC) module containing a Communication Line Adapter (CLA).

The SFEP/MLCC/CLA to SCC (conceptual demonstration model) interface would probably require a specially engineered interface unit. The interface unit is expected to perform functions such as ID code detection and answer back, data formatting, and any special interface requirements which are not compatible between the units. One line of the CLA (FDX at 9600 bps) is expected to support communications between the SFEP and SCC. The final portion of the functional evaluation unit would consist of a telephone communication link between the SCC and a remote terminal (TTY 40 containing an SCC which replaces the station controller module).

Task B3 Development of a SFEP/SCC Functional Evaluation Interface Unit

The purpose of this task will be to support the optimum design study (Task B5) through evaluation and performance testing with hardware mechanized in the most economic methods available. T&E software generated for this task shall provide the basis for future software development in an optimized prototype unit.

This task consists of developmental efforts in the areas of hardware, software and test plans for initial check-out activities. The hardware portion of this task will consist of efforts involved in obtaining GFE and off-the-shelf equipment, design and fabrication of custom interface elements as defined in Task B2.

The software portion of this task will consist of test and evaluation type software to be used by the SFEP and its interface controller module. Initially, the SFEP will be treated as a Secure Communications Processor (SCOMP) for stand-alone testing. Where possible, the software routines generated for this configuration shall be implemented in the final Multics prototype. The test portion of the software shall permit testing and verification of all possible functions which support the

communications interface between the SCOMP and a remote terminal. The evaluation software will be developed around techniques determined in Task B2 to allow measurement of performance and other data required of the Functional Evaluation Unit.

A test plan will be generated to describe the manner in which checkout and evaluation tests will be conducted and to describe any measurement techniques required during developmental testing.

Task B4 SFEP/SCC Functional Evaluation Unit and Multics Integration

The tested Functional Evaluation Unit from Task B3 will be integrated with the Multics system for final tests to support Task B5. To achieve this integration, a 6000/Series 60 Interface Unit is required to provide the SFEP/Multics interface. According to current development schedules, this task could possibly be combined with integration testing of the 6000/Series 60 Interface Unit. Included in this task are the following subtasks:

* Development of T&E Multics software to support interface with the SFEP.

* Development of T&E SFEP software using previously generated subroutines (Task B3).

* Development of a test plan which describes the test/evaluation data required, test configuration, and measurement techniques.

The output from this task will be a final performance and evaluation report which will contain all pertinent information derived from Task B3 and the results of the integration tests.

Task B5 SFEP/SCC Optimum Interface Design Study

The purpose of this task is to optimize a SFEP/SCC interface design which eliminates all dual or redundant functionalities and interfaces while still supporting the secured multiline requirements of the SCC. To be considered in this study task will be possible configurations which incorporate the optimized functions in the SFEP, SCC, or a combination of both. The final configuration will be the result of trade-off decisions in the areas of cost, performance, modularity and the ability to meet requirements previously determined in Task B1.

Major items to be considered, but not limited to, in the optimized interface design study are:

* Security aspects and validation

* TEMPEST compatibility

* Reduction of multiple interconnecting interface types (i.e., RS232 and MIL-STD-188C low-level).

* Reduction of redundant multiplexing functions between multiple communication lines of the SCC and the SFEP bus.

* Overall performance requirements

* Combined line controller functions. The line controller function should minimize the level of SFEP software support while being flexible enough to accommodate various line protocols.

The output from this task will be a detail specification (Part I) for an optimized interface unit, preliminary plans for T&E software (for stand-alone and integrated tests), and definition of possible test equipment.

Task B6 Development of an Optimized Prototype Unit

This task will consist of developmental efforts for an optimized prototype unit in the areas of hardware, software and testing directed by the results of the optimization study task. This task will include the following items:

* Design and fabrication of one prototype unit.

* Procurement of materials

* T&E software development for stand-alone tests (i.e., without central system).

* Test equipment development for stand-alone tests (including definition of GFE and off-the-shelf equipment).

* Test configuration/procedure development for stand-alone tests.

* Design verfication/evaluation tests for a stand-alone configuration.

* Demonstration tests with the SCOMP/SCC linked to a remote terminal with local telephone lines.

* TEMPEST qualification tests to demonstrate compliance with the TEMPEST requirements defined in Task B5.

In addition to the hardware and software generated, a final detail specification (Part II) will be written to complete the activities of this task.

Task B7 Optimized Prototype Unit Integration with Multics

The purpose of this task is to integrate the prototype interface unit with the SFEP, SCC and the Multics host central system to prove the SFEP/SCC interface design.

To achieve this integration, a 6000/Series 60 interface unit and support peripherals will be required. According to current schedules, this task could possibly be combined with the integration testing of the 6000/Series 60 interface unit. Included in this task are the following subtasks:

* Development of T&E Multics software to support the SFEP/Multics interface.

* Development of T&E SFEP software

* Development of an integration test plan describing the test/evaluation data required, test configuration, and measurement techniques.

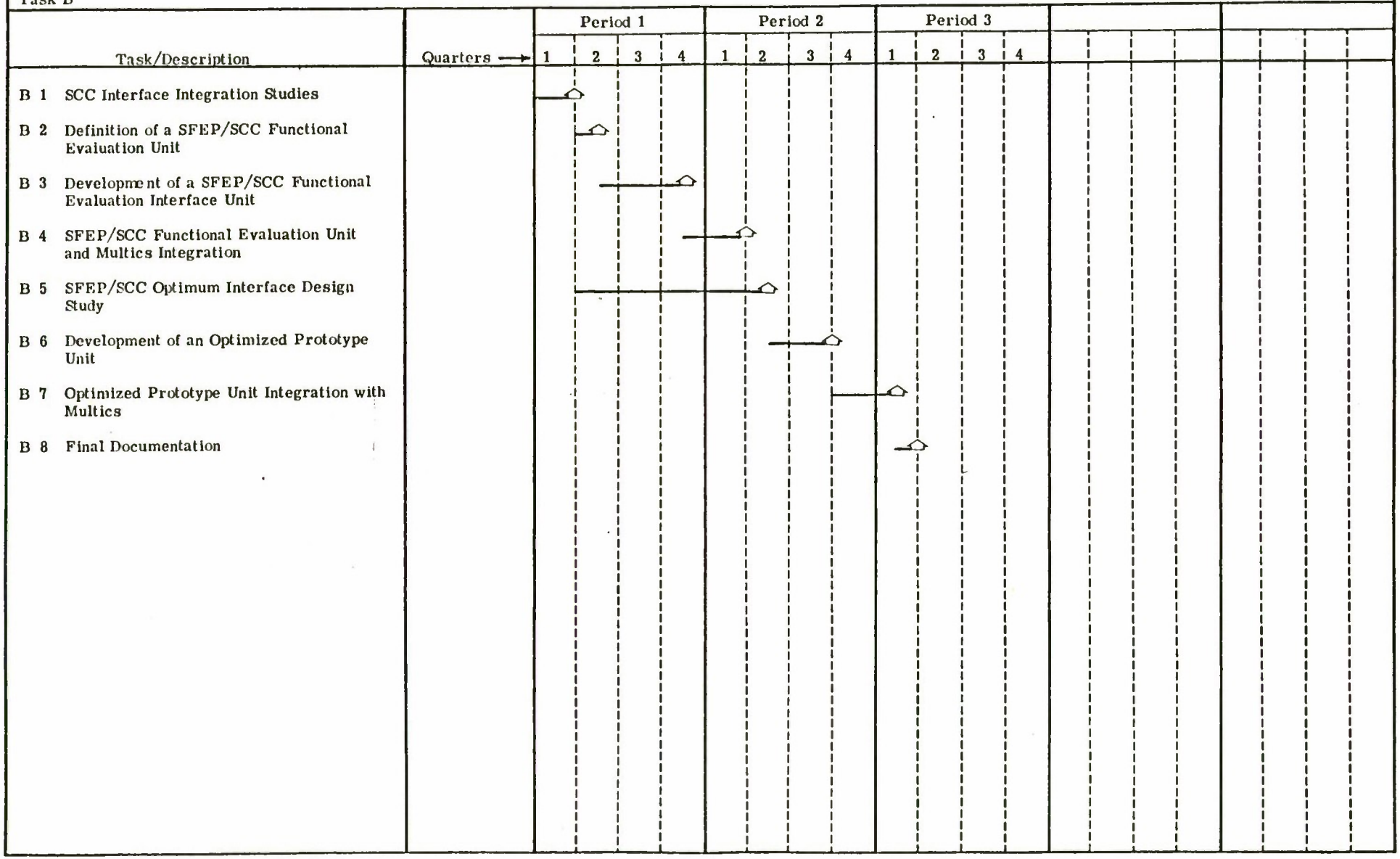
If Task B4 has previously been implemented, portions of the above subtasks will already have been developed.

Task B8 Final Report

The objective of this task is to unite all data, rationale and trade-offs obtained from all the previously defined tasks into a single technical report.

PROJECT GUARDIAN
SFEP/CRYPTO GRAPHIC MULTIPLEXER INTEGRATION

Task B



MANPOWER
MANMONTHS/QUARTER

PROJECT GUARDIAN
SFEP/CRYPTO GRAPHIC MULTIPLEXER INTEGRATION

TASK B

| Task/Description | Quarters → | Period 1 | | | | Period 2 | | | | Period 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| B 1 SCC interface integration Studies | 3.0 | 3.0 | | | | | | | | | | | |
| B 2 Definition of a SFEP/SCC Functional Evaluation Unit | 1.5 | | 1.5 | | | | | | | | | | |
| B 3 Development of a SFEP/SCC Functional Evaluation interface Unit | 16.5 | | 1.5 | 8.5 | 5.5 | | | | | | | | |
| B 4 SFEP/SCC Functional Evaluation Unit and Multics integration | 7.5 | | | | 1.5 | 6.0 | | | | | | | |
| B 5 SFEP/SCC Optimum interface Design Study | 12.0 | | 1.5 | 2.5 | 3.0 | 4.5 | 0.5 | | | | | | |
| B 6 Development of an Optimized Prototype Unit | 24.0 | | | | | | 8.0 | 16.0 | | | | | |
| B 7 Optimized Prototype Unit integration with Multics | 8.5 | | | | | | | | 5.5 | 3.0 | | | |
| B 8 Final Documentation | 1.5 | | | | | | | | | 1.5 | | | |

## Supporting Effort C: Secure Office Terminal

### Task C1.0 INTRODUCTION AND OVERVIEW

This supporting effort encompasses the hardware and software activities required to integrate a Secure Office Terminal into the Secure Front-End Processor (SFEP). Integration of Secure Office Terminals into the (SFEP) is required as a part of the overall effort to develop a multilevel Secure Multics for DOD applications.

The hardware integration of Secure Office Terminals into the SFEP may be almost totally effected via existing subsystems and modules. However, this may not be the most cost-effective approach. This plan encompasses the activities required to develop a cost-effective approach.

A prime approach for integration of Secure Office Terminals into the SFEP is to utilize the Air Force/ESD Secure Communications Controller (SCC). (1) This approach utilizes the Custom Interface Adapter to be developed under Task 4.8.3 of the Project Guardian Statement of Work. (2)

The software modules required for integration of Secure Office Terminals into the SFEP will be developed under the software development activities of the Guardian Program. This includes the software required for the lower levels of application-specific interface protocol and device drivers. The only software described here is for test and evaluation purposes.

The planned activities for accomplishing the integration of Secure Office Terminals into the SFEP are defined in Section C4.0. The attached task schedule provides an estimate of the manpower required to perform these tasks.

### Section C2.0 TASK PLAN

#### Task C1 Integration Requirements Analysis

This task will define the end-to-end hardware and software elements required to integrate Secure Office Terminals into tne SFEP.

---

(1) Secure Communications Controller - Project 7820/01/01.

(2) "Statement of Work for Secure Multics Design, Development, and Certification", 21 November 1975 (Revised). Contract No. F19628-74C-0193

Major issues which will guide the integration requirements analysis are as follows:

1. Communications Security (COMSEC).

   Including encryption requirements; multiplexed-encryption; single-key/multi-key; consumables labeling and marking.

2. Concurrent Multilevel Operation.

   Including multilevel and unilevel terminals, logical security enforcement; line handlers; user authentication; and device validation.

3. Functional Modular Partitioning.

   Hardware and software.

4. Error Control.

   Including initialization; cyclic redundancy codes; echoing and parity.

5. TEMPEST.

   End-to-end and integration requirements.

6. Secure Data Management.

   Including physical access enforcement; audit-trails; surveilance and external alarming.

Task C2 Integration Configuration Derivations

This task will trade-off and evaluate various end-to-end configurations and derive a configuration which maximizes use of standard hardware and software functional modules.

Software Partitioning

The major part of the Multics and SFEP software will be developed under major Tasks 1 and 3 of this Integration Plan. This software encompases only that additional software, firmware, and, perhaps, hardware which is required for integration of Secure Office Terminals into the SFEP.

For example, due to the advent of "smart" terminals, multiplexed cryptographic devices and Multi Line Channel Controllers, which are microcomputer-driven, much of the lower levels of interface protocol may be handled by these devices. These should be designed to be transparent to the higher levels of protocol with the lower level devices appearing as strictly "logical" devices.

Partitioning recommendations for these "lower-levels" of protocol will be developed.

Hardware Partitioning

A generic functional partitioning based on existing subsystems shows that Secure Office Terminals may be integrated with the SFEP via existing standard subsystems and modules.

An alternate partitioning based on the ESD/DCW "Secure Communications Controller" (SCC) eliminates many of the COMSEC devices.

Integration Optimization

The first hardware subsystem partitioning, while perhaps cost-effective through its use of standard modules, indicates that hardware redundancy may be present in interface modules, TEMPEST filter, cryptographic and channel control areas.

Similarly, the second partitioning shows that hardware redundancy (to a lesser degree) may be present in the interface module, TEMPEST filter, time division multiplexor and channel control areas.

This subtask will trade-off alternate implementations which may reduce the hardware and software redundancies. For example, the microprocessor controlled MLCC includes channel multiplexing and line adapter functions. Further integration to include the cryptographic function may be cost-effective.

Selected Configuration

The selected configuration will be based on trade-offs among the following three prime candidate approaches:

1. Utilizing off-the-shelf encryption devices and multiplexors.

2. Utilizing the ESD developed SCC.

3. Utilization of custom designed modules which provide a greater degree of functional integration.

The output from Task C2 will be a Technical Note defining the Integration Configuration derived including block diagrams and hardware/software functional partitioning requirements.

Task C3 Functional Evaluation

Task C3.1 Performance Analysis

This task encompasses a top-level performance analysis based on parametric estimates of subsystem performance.

This will include the performance impact due to security constraints such as:

1. Cryptographic and keying functions.

2. User authentication and device validation.

3. Secure subsystems performance.

Task C3.2 Test Bed System Design

Upon completion of the RNML, SPM and 6000/Series 60 IU integration tests, the resultant SFEP may be utilized for additional evaluation testing. The purpose of this subtask is to procure subsystem modules for a test bed system. This test bed will be configured essentially totally from off-the-shelf equipment and will be utilized to further evaluate end-to-end integration of secure office terminals into tne SFEP.

This will include multiplexed cryptographic subsystems such as the ESD/DCW "Secure Communications Controller" or standard multiplexors and KG's such as those utilized at the AFDSC installation. Also included is a secure office terminal such as tne Secure Teletype Model 40 or the Secure Honeywell Model 7700 VIP.

The test bed system design selection will be based on off-the-shelf modules which replicate the functional partitioning defined under Task C2 of this task plan.

The output from this task will be a tecnnical note which defines the test bed system and delineates all subsystems which must be procured or provided as GFE.

Task C3.3 Test and Evaluation Software Design

This software includes basic software required to evaluate the Secure Test Bed system. This test software will include existing operating system and application software modules augmented by protocol and device driver modules peculiar to the specific terminal, and COMSEC devices selected for the test bed. Certain Kernel and operating system functions may require simulation at elementary logical levels.

Task C3.4 Integration Evaluation

The Test Bed system and T&E software will be integrated and utilized to demonstrate end-to-end Secure Office Terminal performance and throughput.

If classified COMSEC equipment is utilized, physical security will be provided. The COMSEC equipment will be functionally utilized, however, operational keying will not be employed in order to mitigate or eliminate physical security requirements during the evaluation tests.

The output from this task will be a technical note delineating performance and throughput measures and physical security requirements for the Secure Office Terminal.

Task C4 Final Report

A final technical report which describes all trade-off's performed, all design selection rationale for hardware units, and all integration data will be provided.

PROJECT GUARDIAN
SFEP/SECURE OFFICE TERMINAL INTEGRATION

Task C

| Task/Description | Quarters → | Period 1 | | | | Period 2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| C 1 Integration Requirements Analysis | | | | | | | | | |
| C 2 Integration Configurations Derivation | | | | | | | | | |
| C 3 Functional Evaluation | | | | | | | | | |

MANPOWER
MANMONTHS/QUARTER
Task C

PROJECT GUARDIAN
SFEP/SECURE OFFICE TERMINAL INTEGRATION

| Task/Description | Manmonths Quarters → | Period 1 | | | | Period 2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| C 1 Integration Requirements Analysis | 6.0 | 3.0 | 3.0 | | | | | | |
| C 2 Integration Configurations Derivation | 4.0 | | | 2.0 | 2.0 | | | | |
| C 3 Functional Evaluation | 16.0 | | | | | 12.0 | 4.0 | | |

## Supporting Effort D: Security Audit and Surveillance

### Section D1.0 INTRODUCTION AND OVERVIEW

This effort involves extending the capabilities of the secure Multics system to include a security audit function and a security surveillance function. Both of these functions will be under the control of the System Security Administrator.

A security audit is an after-the-fact examination of system logs and records. Its purpose is to maintain accountability in accordance with the policy of the DoD Information Security Program. The audit will detect unauthorized activities within the system, detect attacks of the system (which are defined to fail in a certified secure system), and oversee the use of the system. Events of interest to a security audit may include the recording of incorrect passwords, attempted unauthorized access to segments, logins by selected persons, logins from selected terminals, opening of selected files, and all denials of access due to failed security checks so that a security administrator could determine past events. Since the security audit is concerned with records of events, it presents little added exposure for the system and may be fairly easy to design and implement.

A security surveillance function is an active, on-line examination of activities on the system, examination of interesting events as they occur and continued monitoring of the effectiveness of the security controls (i.e. subverter). The purpose of security surveillance is to monitor for unauthorized activities, penetration attempts, etc., while they are in process. Security surveillance may include having the system issue alarms when selected persons or selected terminals login, when selected files are opened, or when selected security checks are failed. The function may also involve the monitoring of selected terminal interchanges. An active process, sometimes called a "subverter" process, may be used to continuously test the system's security mechanisms and to issue an alarm if any security mechanism is not functioning properly. Provision of the security surveillance function may result in special case situations for security checks, in considerable complication of the kernel mechanisms, and in the possibility for a substantial added exposure for the system. As a result of these considerations, security surveillance may be a difficult function to design and implement.

### Section D2.0 TASK PLAN

#### Task D1 Study and Evaluation

This task will first extend the study of the security audit and surveillance functions begun for the AFDSC and reported in a

tecnnical report. (1) As in the earlier study, this must be a joint effort between the Air Force and Honeywell, with each contributing from their own area of expertise. In particular, the study will have to determine the extent to which security audit and surveillance is required in a certified secure system. The study team will evaluate the costs of the desired features, considering both costs in system performance and in possible increased exposure.

The requirements for the security audit and surveillance functions will be specified in a technical report. This report will then be subjected to a comprehensive design review to ensure that the specified functions meet the security requirements of the Air Force.

Task D2 Functional Specifications

The top level specifications of the preceding Task D1 will be sorted into kernel and non-kernel items. Those items that must be implemented in the security kernel will be integrated into the specifications developed under major Task 1. These functions will then extend and join the Multics kernel development task. These features must be subjected to the full discipline of the Multics kernel development. They should be targeted for inclusion with Multics Prototype Number 3. This requires that the program be started essentially at the start of major task 1. Non-kernel items will be described by a functional specification which will be issued as a technical report.

Task D3 Implementation

The items within the kernel will be implemented by an extension of major Task 1. The non-kernel items will be implemented under this task. A preliminary manual for the System Security Administrator will be prepared to assist in the use of the audit and surveillance functions.

The security audit and surveillance functions will be demonstrated in conjunction with Multics prototype Number 3.

---

(1) J. Whitmore, et al, "Design for Multics Security Enhancements", ESD-TR-74-176, Honeywell Information Systems, Inc., Cambridge, MA, December 1973.

PROJECT GUARDIAN
SYSTEM AUDIT AND SURVEILLANCE

Task D

| Task/Description | Quarters → | Period 1 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | | | | | | | | | | | |
| D1 Study and Evaluation | | ◇ | | | | | | | | | | | | | | | |
| D2 Functional Specification | | —— | ◇ | | | | | | | | | | | | | | |
| D3 Implementation | | | | —— | ◇ | | | | | | | | | | | | |

MANPOWER
MANMONTHS/QUARTER

PROJECT GUARDIAN
SYSTEM AUDIT AND SURVEILLANCE

Task D

| Task/Description | Manmonths Quarters → | Period 1 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | | | | | | | | | | | |
| D1 Study and Evaluation | | 4.0 | | | | | | | | | | | | | | | |
| D2 Functional Specification | | 2.0 | 6.0 | | | | | | | | | | | | | | |
| D3 Implementation | | | | 6.0 | 6.0 | | | | | | | | | | | | |

## Appendix B

## Air Force ESD Comments

Page 28 -- Line +2 (second line from top of page). Definitive goals as well as products need to be defined for the program correctness proofs task.

Page 28, Task 1.28 -- This task avoids discussion of a potential interface with the Autodin II message switching network. The design for a network should be modular so as to be adaptable to Autodin II.

Page 29, Task 1.29 -- This task does not appear to be a technical task but rather a management task. Honeywell should try to utilize existing trained and cleared personnel whenever possible.

Page 31-34, Tasks 1.33-1.40 -- Although these tasks address the analysis of security vulnerability due to hardware failure, there is no mention of developing techniques, procedures or software for minimizing the occurence of the failures found to seriously affect the operating of the certifiable Multics system.

Page 33, Task 1.37 -- The objective of this task is not clear.